AD-A115 757 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 17/2
AN INVESTIGATION OF THE APPLICATION OF VOICE INPUT/OUTPUT TECHN--ETC(U)
MAR 82 T R MALARKEY
UNCLASSIFIED NL

1 OF 2
A
115757

2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INVESTIGATION OF THE APPLICATION OF VOICE INPUT/OUTPUT TECHNOLOGY IN THE COINS NETWORK CONTROL CENTER

by

Thomas R. Malarkey

March 1982

Thesis Advisor: G. K. Poock

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A115757	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Investigation of the Application of Voice Input/Output Technology in the COINS Network Control Center		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas R. Malarkey		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1982
		13. NUMBER OF PAGES 156
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Automatic Speech Recognition
Voice Output
Computer Networking
Computer Simulation
COINS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

During the 1980s we will continue to see an increased use of distributed computer networks. Although network usage has been found to be effective in a wide variety of applications, continued network expansion heightens the need for effective management to achieve optimum performance, reliability, and security of network operation. Advances in network management have not kept pace with the problems that arise in network operation. The Community On-Line Intelligence System (COINS) is a packet-switched network connecting organizations in the U.S. intelligence community. COINS exhibits many of the difficulties faced by large networks. It is ironic that networks have made advanced technology available to a large number of users, yet the use of advanced technology to assist network management has been relatively limited. This thesis will study the COINS Network Control Center (CNCC) and explore ways that Voice Input/Output (VIO) Technology can be used to improve the day-to-day management of network operations.

An Investigation of the Application of Voice Input/Output Technology in the COINS Network Control Center

by

Thomas Raphael Malarkey
B.A., Pennsylvania State University, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(COMMAND, CONTROL AND COMMUNICATIONS)

from the

NAVAL POSTGRADUATE SCHOOL
March 1982

Author: [illegible]

Approved by: [illegible]
Thesis Advisor

[illegible]
Second Reader

[illegible]
Chairman, C3 Academic Group

[illegible]
Academic Dean

ABSTRACT

During the 1980's we will continue to see an increased use of distributed computer networks. Although network usage has been found to be effective in a wide variety of applications, continued network expansion heightens the need for effective management to achieve optimum performance, reliability, and security of network operations. Advances in network management have not kept pace with the problems that arise in network operation. The Community On-Line Intelligence System (COINS) is a packet-switched network connecting organizations in The U.S. intelligence community. COINS exhibits many of the difficulties faced by large networks. It is ironic that networks have made advanced technology available to a large number of users, yet the use of advanced technology to assist network management has been relatively limited. This thesis will study the COINS Network Control Center (CNCC) and explore ways that Voice Input/Output (VIO) Technology can be used to improve the day-to-day management of network operations.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to express my thanks to my thesis advisor Professor G. K. Poock for introducing me to the world of voice technology; to Professor Alvin Andrus for passing on his knowledge of computer simulation and for serving as second reader; to Ellen Roland for her practical advice; and to Mr. George G. Hicken, the CCINS Program Manager for support and information.

I would also like to thank my wife Joan for help, encouragement, and especially understanding.

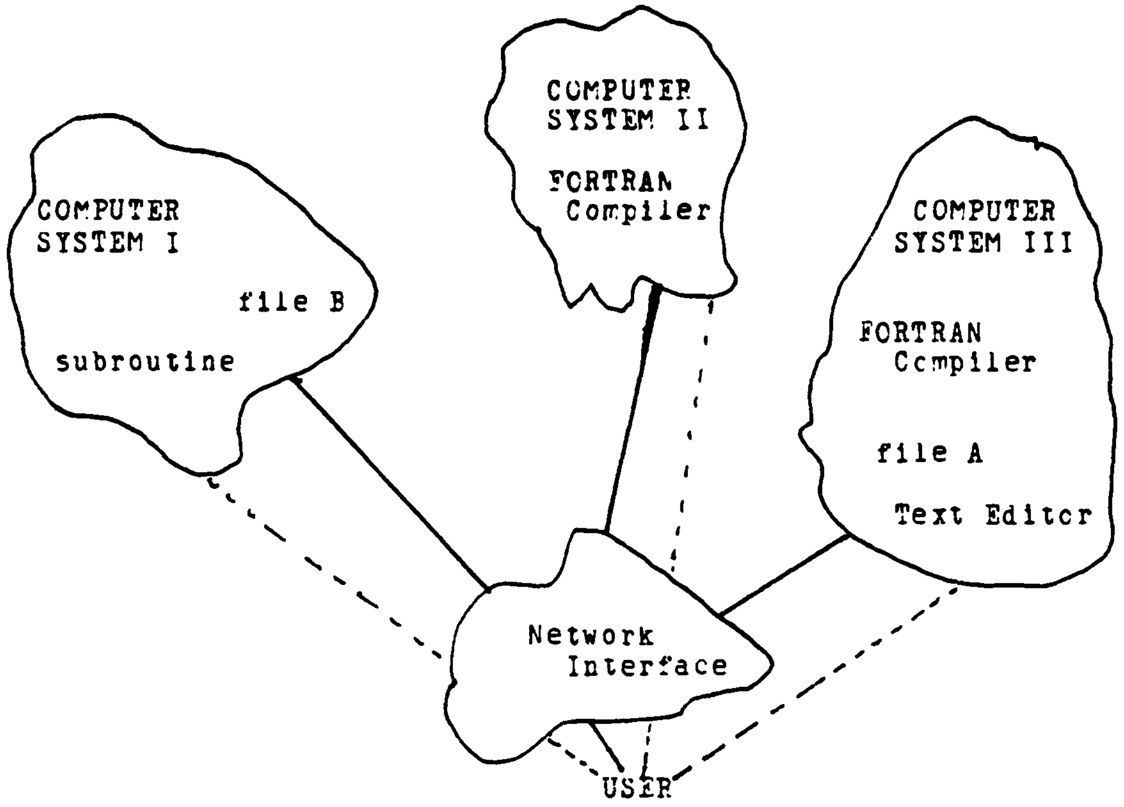
I. INTRODUCTION

A central technological theme of the 1970s was the coupling of two technologies: computing and communications. Dramatic reduction in the cost of digital-technology and a less dramatic, but still sizeable, expansion of data transmission capabilities have spurred the introduction and growth of several geographically distributed computer networks. Pioneered by a combination of government research and private initiative, computer networks are becoming increasingly available for a wide variety of applications.

The essential characteristic of a computer network (or more properly, a computer communications network) is that the user views the network as a collection of computing resources that provide a range of diverse services and capabilities [Ref. 1: p. 1010]. These computational resources are accessible to users through a network architecture that is more or less transparent. In most of today's networks, the user must establish a logical connection to one of the network computers and use a set of host-dependent commands to access and manipulate data. Networks of the future are likely to to become increasingly transparent in the sense that the user will view the entire network (and connected networks) as a single black box that provides a set of information processing resources that can be accessed by a common command and query language. Figure 1 illustrates the user's view of a present-day computer communications network.

Much of the work in advancing network technology has been spearheaded by The Advanced Research Projects Agency (ARPA) of the Department of Defense (DoD). The ARPANET, an experimental network employing packet-switching technology, has been operational since 1969. This network connects over fifty host computers at more than forty different locations. This geographical dispersion is accompanied by considerable diversity in the types of computers that serve as host machines. In addition to its own networking activities, the ARPA technology has influenced a number of other distributed information processing projects both within DoD and in other government agencies. One of these efforts was the Community On Line Intelligence System (COINS) which interconnects on-line information storage and retrieval systems located at a number of locations within the U.S. intelligence community.

As with many areas of advanced technology, the technical problems of computer networking have been solved fairly easily; the problems of network management and control have been a good deal less tractable. It is ironic that the control mechanisms for computer networks, which embody advanced ADP and communications technology, are generally rather primitive. The COINS Network Control Center (CNCC), for example, has several serious limitations in hardware,



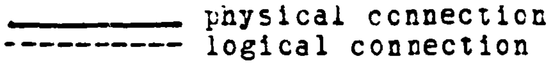


Figure 1. User's View of a Computer-Communications Network
(Reprinted from Reference 1)

software, and in operator assistance. The purpose of this thesis is to study the existing CNCC and to explore ways that Voice Input/Output (VIO) technology might be used to help the CNCC operators better manage the network. To assist in this process, we have designed and implemented a computer simulation of the CNCC to help us study the applicability and feasibility of voice technology.

We will first describe the COINS Network and the CNCC. We will then summarize some of the advances in VIO technology over the last several years and analyze the extent to which the technology is likely to improve the CNCC operation. We will then describe the design of the CNCC computer model and present some preliminary conclusions that were obtained by using it. Lastly, we will discuss implementation strategies, costs, and potential pitfalls associated with installing VIO in the CNCC.

II. THE COINS NETWORK

A. BACKGROUND

The Community On Line Intelligence System grew out of the need for U.S. intelligence agencies to share information. COINS I originated as a star network with a message-switching computer in the star's logical center at NSA. In the early days of COINS I operation, the switch computer served as a de facto Network Control Center because all traffic was routed through it enabling it to monitor the status of the network [Ref. 2: p. 3-1]. COINS II, developed in the mid-1970s, uses the ARPA packet-switching technology to connect its host computers. There is, therefore, no longer a central message-switch to perform the network control function.

The current network consists of several host systems (some of which are served by PDP-11 front-end processors). Hosts are connected to the network by Interface Message Processors (IMPs) interconnected by wideband communication lines. The IMPs form a reliable and secure communications subnetwork which uses a variety of techniques, such as adaptive routing, to ensure network performance and availability.

During a typical COINS operation, a user at a terminal connected to some host (or alternatively to a Terminal

Access System (TAS)) enters an interrogation request for retrieval of information that resides at some other network host. The retrieval request is passed from the host to its IMP where it is divided into packets approximately 1000 bits long. These packets are independently routed through the network until they arrive at the destination IMP where they are reassembled into the original interrogation message. After reassembly, the destination IMP passes the request to its host computer for action. The destination host processes the interrogation by retrieving and formatting the requested data from its files and passing the answer, through the network, back to the requesting terminal [Ref 3: p. I-3]. COINS hosts provide a variety of query languages, file structures, network protocols, and terminal types. Additionally, The COINS Project Management Office (COINS PMC) actively develops hardware and software components to facilitate the smooth and rapid exchange of information between users on diverse hosts. The Technology Transfer Research Facility (TTRF), a part of the COINS PMO, is actively involved in projects such as the DARPA-sponsored Man-Machine Relationship Program and the development of a multiple retrieval language translator (called ADAPT) which is intended to serve as a common user interface.

B. NETWORK CONFIGURATION

Conceptually, the COINS II network can be viewed as consisting of four independent layers or rings as shown in Figure 2. The Host ring consists of relatively large computers that support on-line data bases and provide direct analytic support to intelligence analysts. Examples include the SIGINT On-line Information System (SOLIS) at NSA and the UNIVAC-1110 system at the National Photo Intelligence Center (NPIC). The Access processor Ring consists of various computer systems that allow a user to access COINS data bases without going through a host. Included in this layer are the Terminal Access Systems (TAS) which are roughly analogous to very high-powered TIPs in ARPANET terminology; Front-End Processors (FEPs) which connect hosts to the communications subnetwork; and Gateways, which serve as interfaces between COINS II and other networks. The IMP ring is the communications subnetwork based upon ARPANET philosophy and technology. The T1 (tetrahedron) ring is that activity that links the IMPS together and maintains the secure communications of the COINS II network [Ref. 4: p. 4-5].

In Figure 3, which portrays the current COINS II topology, the independent layers are denoted by different symbols: hosts are represented by rectangles; access processors by triangles; and IMPS by circles. Figure 3 highlights the fact that COINS II is a true distributed

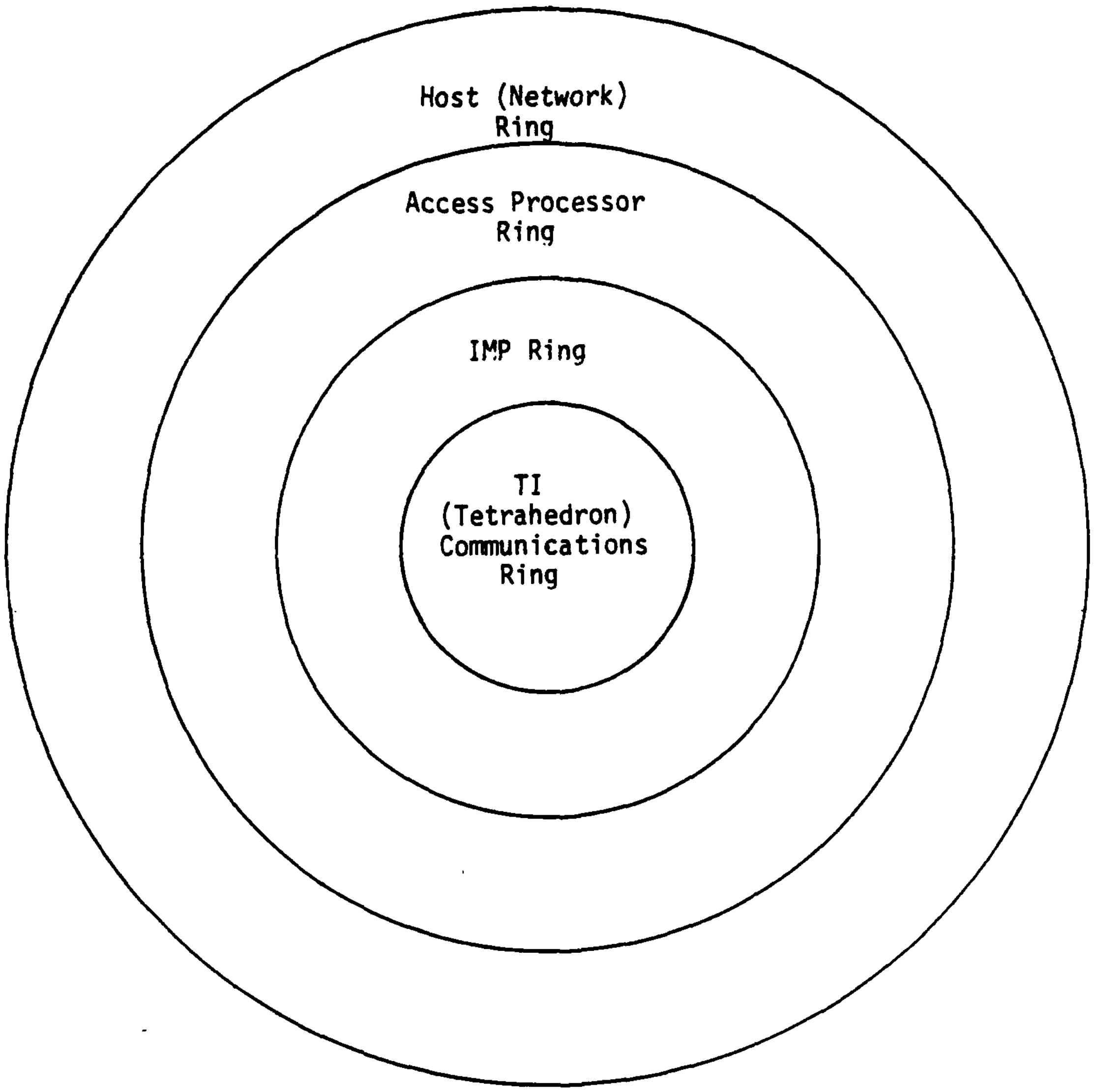


Figure 2. Conceptual View of Coins II (from Reference 2)

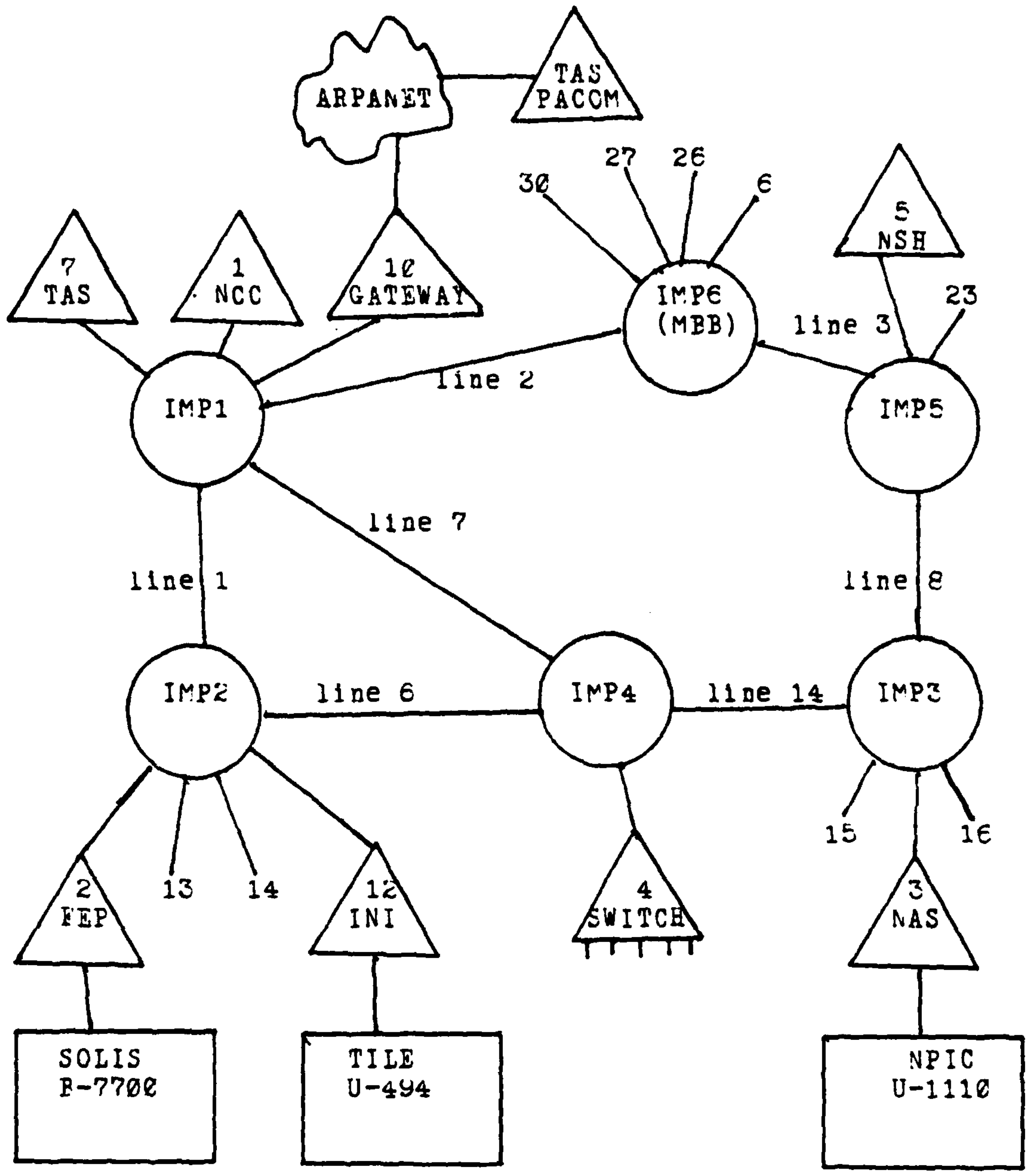


Figure 3. COINS II Network Reference Map
(from Reference 3)

network with functions allocated both to individual layers and to individual components on the same layer. This provides greater reliability and flexibility since the network can continue to operate in the face of host, IMP, or communication line failures. A key feature of the network is the use of an adaptive routing algorithm to pass messages from IMP to IMP. Each IMP is connected to at least two neighbor IMPs. At the time an IMP receives a message from its host, it decides the proper route to use to deliver the message to its destination based on the current state of the network. For example, to route a message from IMP 1 to IMP 2, there are three lines from which to choose. Normally, the direct line (line 1) would be the best approach. If IMP 1 detects that line 1 is either down or heavily loaded, it can choose to route the message indirectly over two other possible routes.

C. NETWORK MANAGEMENT AND CONTROL

Management of a computer network encompasses all facets of operation including long-range planning, training, staffing, and budgeting. In this paper, however, we will be primarily concerned with day-to day management and control of network resources. The chief prerequisite for effective operational control is the ability to rapidly diagnose and respond to failures in the network. These functions are performed by the COINS Network Controller who uses the COINS

Network Control Center (CNCC) computer to assist in the monitoring and control of the network. The CNCC and the Controller will be discussed in greater detail in the next section. Before doing so, however, we will point out some of the general problems of controlling a distributed network and discuss some of the longer range plans for CCINS network management.

The COINS communications subnetwork can be considered a distributed computation system [Ref. 3: p I-5], since message/packet processing, network status monitoring, and trouble reporting are performed independently by each IMP. This distribution of function, while ensuring better overall reliability and service, does complicate the problem of detecting and correcting problems with network components. Geographical distribution further aggravates the control problem. The CNCC was established to address the problems of distributed control. It serves the multiple functions of continually monitoring and diagnosing problem areas, coordinating corrective measures, and providing a central point to which users may direct inquiries and complaints. Each IMP is programmed to examine itself and its interfaces periodically and report the results to the CNCC. The CNCC collects these (possibly conflicting) IMP reports, determines the most likely true state of the network, and suggests the required repair activity in the event of reported failures.

The COINS PMO is planning to implement a fully automated on-line system for the collecting, editing, analyzing, and reporting of network information. The information will be used to monitor the operations, performance, and utility of the COINS Network. This effort, known as the COINS Network Management System (CNMS), is planned to be implemented in stages over the next five years. This paper is intended to complement that effort by exploring some remedial steps that can be taken to improve the performance of the existing CNCC that later could be incorporated into the enhanced CNMS.

III. THE COINS NETWORK CONTROL CENTER

A. CONFIGURATION AND FUNCTIONS

As noted above, the CNCC maintains a dynamic picture of the status of all network components (IMP's, FEPs, INIs, and lines) and is thus able to direct recovery procedures in the event of component failures. The CNCC machine, a Honeywell-316 computer, is attached to the subnetwork by means of a local host connection to the network's master IMP (IMP 1). The operation of the remote IMPs can be controlled either from the CNCC or directly from the Master IMP. The Control Center operator is responsible for both machines.

The CNCC uses two teletypes (known as the Logger TTY and the Summary TTY) as primary output devices. The system uses the Logger to print event-oriented messages as they occur. Logger messages are often concerned with alert information pertaining to the status of network components and will frequently require action by the Network Controller. The CNCC has an alarm box wired into the logic of the Logger TTY which keys on the presence of a sentinel character (ASCII control-G or BELL) in a Logger message. Thus, by changing the text of the Logger messages, any message can be made to set off the alarm box and alert the operator. The Summary teletype is used to print periodic reports and also serves as the primary command input device for the CNCC operator.

The CNCC can also communicate with other COINS hosts such as the UNIVAC 494 host at NSA. These host-to-host communications are used primarily to transfer network status and summary files for further processing and reporting. One of the fundamental objectives of this thesis is to evaluate the use of voice input technology to perform the input function of the Summary TTY and voice output technology to perform the action-alert function of the Logger TTY.

The CNCC program is driven by three sources: (1) reports from the network IMPs, (2) a real-time clock, and (3) input commands typed on the Summary TTY by the operator.

Each IMP sends two reports to the CNCC computer at least once every 52 seconds. The first of these, the status report, provides information about the operational condition of the IMP itself, modems, and connected hosts; the second, the throughput report, provides data throughput information in terms of both words and packets for all modems and hosts. If these reports suggest a change in status of any network component, the CNCC will output a message on the Logger TTY that may require action by the CNCC operator. The statistical information is accumulated and output on the Summary TTY either automatically, in response to an interrupt from the realtime clock, or on demand, in response to a command from the Network Controller.

The CNCC program accepts commands typed by the operator on the Summary TTY and performs a variety of functions to monitor and control the network configuration. These functions include reloading or restarting an IMP, testing a host or modem interface, or selecting one of the statistical summary reports for display.

B. ROLE OF THE NETWORK CONTROLLER

The COINS Network Controller is responsible for overall operation of the network. It it important to note that, although we often refer to the Network Controller as "the operator", his duties are considerably broader than those of a computer operator in a traditional ADP center [Ref. 5]. The Network Controller's role includes the coordination and monitoring of network performance, production of network status reports, diagnosis of failures of network components, and direction of recovery procedures. Given the distributed nature of the COINS II network, the Controller has first-order responsibility for network operations. The major duties of the Network Controller are summarized in this section [Ref. 3: pp. I-10 - I-11].

The Network Controller is responsible, first of all, for operating the CNCC complex. This includes the loading, starting, and normal operation of both the CNCC and the Master IMP programs. Although in this thesis we are primarily concerned with the CNCC H-316 computer, the CNCC Network Controller is responsible for a much larger hardware suite. Other computers in the CNCC complex include the

PDP-11/70 Terminal Access System (TAS) that interfaces 32 user terminals to the network; the Network Service Host (NSH), another PDP-11/70 system used for research and development and network software support; and several other special-purpose computer systems [Ref. 6: p. 2-4]. It is planned that ultimately the Network Controller could monitor all these systems from a single integrated operator console.

The network monitoring activity consists mainly of observing the CNCC Logger reports and properly interpreting error and failure indications. Duties include coordinating network status with the COINS PMO, scheduling of special network tests, gathering performance and throughput statistics, verifying the programs at the network IMPs, and the coordinating any needed maintenance support. The Network Controller is also responsible for monitoring the CNCC hourly summary reports, requesting additional reports if necessary, and off-loading report files and network statistics to the TIPS1 system at NSA for further processing and analysis.

The most important duties of the Network Controller are those that affect the day-to-day operation of the network. These include diagnosing network or user problems, conducting network tests to measure performance, and releasing new software to the network IMPs. The Network Controller's fundamental responsibility is to keep all network components operating at a level that assures users of reliable and timely service. Additionally, he serves as the focal point for questions, problems, and complaints about all aspects of network operation.

C. CNCC INPUTS AND OUTPUTS

1. Summary Teletype Output

The messages printed on the Summary TTY contain network information that has been compiled by the CNCC computer from IMP status and trouble reports. Summary reports fall into three general categories: those that provide information on the current status of network components; those that provide accumulated summaries of status information; and those that provide accumulated summaries of host and line throughput[Ref 2: Appendix B]. The Network Controller can select any of these reports by entering commands at the Summary TTY. In addition, accumulated summaries of throughput and status information are output automatically every four or eight hours. The appropriate operator commands are described below.

There are two current status reports: QUICKPRINT and HOST STATUS. The QUICKPRINT, a sample of which is shown in Figure 4, uses one-character codes to describe the current status of all IMPs and lines in the network. This report also displays any unusual situations (e.g., sense switches ON, OVERRIDE ENABLED, etc.) The following status code definitions are used in the QUICKPRINT report:

```
Ø53Ø AUGUST 5 1981  QUICK SUMMARY

IMP
......
1234567890 1234
+<.??...?? ???.
LINE


IMP 1 SS 4 ON
      MSG GEN ON
```

Figure 4. Sample QUICKPRINT Report

?	status unknown
.	up
*	down
+	plus side of line down
-	minus side of line down
>	plus side of line looped
<	minus side of line looped
Z	both sides of line looped

The HOST STATUS report gives the current status of all hosts in the network or of only those hosts connected to a specific IMP. A sample HOST STATUS report is shown in Figure 5.

The CNCC accumulates status information on all IMPs and lines on a daily basis. This information is displayed in the STATUS report. Status is reported in terms of 15 minute segments such that an entry is printed for a time period only if the status of one of the components changed during that interval. The STATUS report employs the same set of codes used in the QUICKPRINT with the addition that, if the status of any IMP or line has changed during the interval, its status will be shown as "X". A sample of the STATUS report is shown in Figure 6. The IMP and line STATUS reports are automatically summarized and output three times a day, at 0800, 1600, and 2400.

The CNCC compiles throughput statistics on the amount of traffic transferred between hosts and over the

```
0532 AUGUST 5 1981  HOST SUMMARY

IMP
......
123456

IMP      1          HOST 0 (VIRT-HOST  1)  UP
                    HOST 1 (VIRT-HOST  7)  DOWN
                    HOST 2 (VIRT-HOST 10)  DOWN

IMP      2          HOST 0 (VIRT-HOST  2)  UP
                    HOST 1 (VIRT-HOST 12)  UP
                    HOST 2 (VIRT-HOST 13)  UP
                    HOST 3 (VIRT-HOST 14)  DOWN

IMP      3          HOST 0 (VIRT-HOST  3)  DOWN
                    HOST 1 (VIRT-HOST 15)  UP
                    HOST 2 (VIRT-HOST 16)  UP

IMP      4          HOST 0 (VIRT-HOST  4)  UP

IMP      5          HOST 0 (VIRT-HOST  5)  DOWN
                    HOST 1 (VIRT-HOST 23)  DOWN
                    HOST 2 (VIRT-HOST 24)  DOWN

IMP      6          HOST 0 (VIRT-HOST  6)  DOWN
                    HOST 1 (VIRT-HOST 26)  DOWN
                    HOST 2 (VIRT-HOST 27)  DOWN
                    HOST 3 (VIRT-HOST 30)  DOWN
```

Figure 5. Sample HOST STATUS Report

```
AUGUST 5 1981  COINS NETWORK SUMMARY 0000-1600

IMP STATUS

TIME    123456

0000    ??????
1030    XXXXXX
1045    ......

LINE STATUS

TIME    1234567890 1234

0000    ?????????? ????
1030    XXX??XXX?? ???X
1045    ...??...?? ???.
1315    ...??...?? ???.
1330    ...??.XX?? ???.
1345    ...??...?? ???.
```

Figure 6. Sample STATUS Report

lines of the network. This information can be displayed in either of two reports: the hourly THRUPUT summary report and the eight-hour DAYSUM report. The formats of these reports are identical; the statistics, however, may differ somewhat depending on the time of the request.

In the THRUPUT report, shown in Figure 7, information is arranged by IMP and within each IMP by physical host address with the virtual host address (VHA) in parentheses. Host throughput is reported in terms of the number of messages and packets sent and received for both inter-site and intra-site traffic. The following symbology is used to denote the information unit and path direction in the THRUPUT report:

M->N	Messages sent to hosts at other network sites.
M<-N	Messages received from hosts at other sites.
P->N	Packets sent to hosts at other sites.
P<-N	Packets received from hosts at other sites.
M->S	Messages sent to hosts at the same site.
M<-S	Messages received from hosts at the same site.
P->S	Packets sent to hosts at the same site.
P<-S	Packets received from hosts at the same site.

Throughput figures less than 32,767 are exact; those between 32,768 and 2,097,151 are accurate to within 3%; and those greater than 2,097,151 are shown as +++++++.

```
AUGUST 5 1981  COINS NETWORK SUMMARY 1400-1500   PAGE 8

          HOST 0  HOST 1  HOST 2  HOST 3
IMP01     (  1)   (  7)   ( 10)
M->N              2693     283
M<-N       698    2962     691
P->N              3439     361
P<-N       698    6168     867
M<-S       140
P<-S       140

IMP02     (  2)   ( 12)   ( 13)
M->N      1655    1765     117
M<-N      1708    1808     122
P->N      3670    3151     385
P<-N      1890    1921     181

IMP03             ( 15)
M->N               281
M<-N               669
P->N               729
P<-N               720

AUGUST 5 1981  COINS NETWORK SUMMARY 1400-1500   PAGE 9

LINE THROUGHPUT

LINE  1: IMP01 TO IMP02  9486  IMP02 TO IMP01  11090
LINE  2: IMP01 TO IMP06  1705  IMP06 TO IMP01   1622
LINE  8: IMP03 TO IMP05  3268  IMP05 TO IMP03   2309
```

Figure 7. Sample THRUPUT Report

2. Logger Teletype Output

a. General

The description of the Summary teletype outputs was relatively straight forward, since the outputs were well-defined and fairly predictable. Such is not the case when we consider the outputs that appear on the Logger teletype. We recall that the Logger TTY is used for alert type messages, many of which require some action by the Network Controller. Appendix B of Reference 2 contains a list of 93 possible output messages that may be sent to the Logger TTY. Many of them are merely informational in nature and are useful primarily for later diagnosis and tracing of events. It is, therefore, neither necessary nor desirable for our purposes to describe them all. Instead, we will present the general form of the Logger messages and describe those that pertain to the overall performance of the network. The messages with which we will be concerned are mainly those that will activate the COINS alarm box to audibly alert the Network Controller. These messages are the ones that would be candidates to be sent to a voice output device. A fuller discussion of Logger messages is found in Reference 3.

All Logger messages have the following general form:

```
<time> <reporting net component>: <event descr>
```

where <reporting net component> is some IMP or line and <event descr> is a brief message describing the specific event. Some examples are:

```
  0745   IMP  1:  VIRT-HOST 10 DOWN
  1349  LINE  8:  ERRORS MINUS 5/6
  1350   IMP  5:  TRAP  (74, 401,  1)
```

In the remainder of this section, we will describe some of the more important Logger TTY messages.

b. BAD HOST DATA CHECKSUM

Each IMP has internal tables on which it bases message routing and delivery decisions. Should one of these tables become disturbed, for example because of a program crash, this message will appear on the Logger at one minute intervals until the table is repaired.

c. HOST N DOWN

The IMP is reporting that physical host N has dropped its ready line. A variation of this message is "VIRT-HOST N DOWN" which provides the same information for a host that is assigned a virtual host number. Because local host operation is not a CNCC responsibility, this message is mainly to provide information that might be used to answer user queries.

d. IMP N DOWN

The neighbors of IMP N have reported that the lines to that IMP are down. The CNCC has decided that the

IMP is not available for network service although it may still be actually running and able to communicate with its local hosts.

e. LINE X DOWN

The IMPS at either end of line X have declared the line dead. The CNCC makes the decision to report the line dead.

f. LINE X DOWN PLUS (or MINUS)

The end of a line which is connected to the higher numbered IMP is defined to be the "plus" side of the line; that connected to the lower numbered IMP is the "minus" side. This logger message is similar to the previous one except that the IMP at only one end of the line is reporting a problem.

g. TRAP (T,A,X)

The reporting IMP has detected an internal anomaly. the T, A, and X values provide further information about the trap. Usually these messages are of interest to network software personnel only; some of them, however, require operator action to clear up the error condition.

h. CRASH (T,A,X)

This message indicates that the reporting IMP has detected a serious program malfunction that caused the program to crash. It has reloaded itself and the program is now up and running. T, A, and X contain register values useful to software personnel diagnosing the failure.

i. VIRT-HOST-TBL X

This message indicates that the serial number of the reporting IMP's virtual host address (VHA) table differs from the master in the CNCC computer. Appropriate operator action is required to make the serial numbers consistent.

3. Network Controller Commands

a. General

As was the case with the Logger TTY messages there are a large number of CNCC operator commands that do not concern us. We will be concerned primarily with those that are used frequently in network monitoring and troubleshooting. These are also the commands that are good candidates for voice input. Commands to the CNCC are preceded by the command sentinel character "?" and consist of a 1-3 character command mnemonic followed, optionally, by additional command-dependent parameters such as filenames, IMP numbers etc. In the discussions that follow, operator entries are in upper case and underlined; prompts and responses by the CNCC are all in lower case. Expressions in angle brackets, e.g. <FILENAME>, are used to represent ASCII strings. The "$" character is used to denote the ASCII escape character (octal 33). Some examples are:

```
?REPort Daysum:
?Broadcast file: <FILENAME>$ imp: <#>$
```

CNCC commands can be grouped into three broad categories: commands used to produce summary reports; commands used to manipulate local CNCC files; and commands used for remote transmission of files between the CNCC and other network components. For a complete list of CNCC commands, see Appendix E of Reference 3.

b. Summary Report Commands

To request a specific summary report, the Controller types ?REP and the first letter of the report type (e.g. D for DAYSUM report). The command is terminated by typing a colon. The following are the commands for the various summary reports:

```
?REPort Quickp:
?REPort Host status:
?REPort Host status imp:<#>$
?REPort Status:
?REPort Thrptsum:
?REPort Daysum:
```

c. Local File Commands

Local file commands allow the operator to perform routine operations on local CNCC files. The following examples illustrate the types of commands available. For a more complete description see Section III of Reference 3.

```
?Print Directory
?Print Total  (disk space available)
?REName (old) <FILENAME>$ (new) <FILENAME>$
```

d. Remote File Transfer Commands

The commands used to transfer files between the CNCC and other components in the COINS network are probably the most important of all the operator commands. They provide the Network Controller with a means of reloading remote IMPs and correcting other network problems. One of the most frequently used commands is the BROADCAST command which has the following general form:

```
?Broadcast file: <FILENAME$ imp: <#>$
```

where <FILENAME> determines the specific action and <#> is the IMP number to which the file will be sent. Numerous examples of the use of this command can be found in Reference 3.

To correct certain types of checksum errors, the Controller uses the RELOAD command which has the following general form:

```
?RELoad file: <IMPSYS ####BIN##>$
modem: <#>$ imp: <#>$
```

where <IMPSYS #### BIN ##> is the name of the latest version of the IMP program and <#> refers to the modem number and the IMP number of a neighbor IMP to the one reporting the checksum errors.

On occasion, the operator wants to transfer a file from the CNCC disk to another COINS host. Often, these are statistical files sent to another host for follow-on processing. This function can be accomplished by the SEND command which has the following form:

```
?SEnd local-file <CNCCFILE>$
to host <#>$ remote file name<REMFILE>$
```

where <CNCCFILE> is the name of the CNCC file to be transferred and <REMFILE> is the name that the file will have in the file system of the remote host.

D. LIMITATIONS OF CURRENT CNCC CONFIGURATION

As currently configured, the CNCC has a number of shortcomings that make it more difficult for the Network Controller to perform the basic functions of network monitoring and control. Although not all of these problems are germane to the current inquiry, it is worthwhile to discuss each of them briefly to put the existing CNCC in perspective.

The most obvious limitation is the relatively primitive hardware in the CNCC. The CNCC computer has only 32K bytes of primary memory and cannot be expanded; this places a severe constraint on its ability to manipulate and analyze network status information. There is insufficient disk storage to save all the raw data that is sent to the CNCC. As a consequence, data must be summarized before it is stored with a resulting loss of information. The CNCC output devices are quite slow and it is possible to lose traffic in peak situations. It is worth noting that a number of COINS hosts have completed major hardware upgrades while the CNCC hardware still represents the technology of the early 1970s [Ref. 2: p. 3-3].

There are similar problems with the CNCC software, both with the analytical routines and with the procedures that interface directly with the operator. Many of the output messages are cryptic and require considerable experience and guesswork to interpret correctly. The input command language is also not very human-oriented. Some of the corrective procedures are cumbersome and prone to error. For example, the procedure for reloading an IMP can sometimes as many as eight fairly complicated steps [Ref. 2: p. 3-6]. Humanized, self-contained output messages and simple macro type input commands would go a long way to simplify the job of the Network Controller.

One of the most serious problems facing the Network Controller is just the number of individual consoles that must be monitored. As noted above, the CNCC operator is responsible for the operation of several computer systems in addition to monitoring the Logger and Summary TTYs. While these systems are all in the same general area, it is not possible to monitor and operate all the systems effectively without a considerable amount of physical movement. One study showed that, in the course of a typical week, the CNCC operator made 803 separate "trips" between various positions and covered a distance of over five miles. More significant than total distance is the fact that the frequent interruptions make it hard for the Controller to maintain effective continuity over all the positions. He will too often be at the wrong console at the wrong time and thus miss some important messages.

These shortcomings cannot be eliminated by a piecemeal approach. Ultimately, the CNCC will have to be replaced by a modern computer system that enhances and simplifies the job of network management. Plans for such an upgrade have been developed by the COINS PMO. A turnkey replacement system is, however, several years away. Advances in Voice Input/Output (VIO) technology over the last several years provide the potential to make some important interim improvements at comparatively moderate cost. In the remainder of this paper, we will take a closer look at voice technology and examine ways that it might be adapted to the existing system to increase the effectiveness of the COINS Network Controller.

IV. VOICE INPUT/OUTPUT TECHNOLOGY

A. GENERAL

As we noted in the previous section, several of the problems faced by the CNCC Network Controller could be categorized as shortcomings in human factors engineering. One way of improving the human aspect of any system is to simplify the procedures used to communicate the intentions of the human operator to the system. Another is to provide a better way of notifying the operator that a significant event that requires his attention has occurred. What we will attempt to do in the remainder of this thesis is explore and highlight ways of achieving these two fundamental improvements to the current CNCC system. On the input side, we will examine the uses of Automatic Speech Recognition (ASR) equipment; on the output side, we will consider the use of voice output techniques, such as speech synthesis. Input and output will be discussed separately in more detail in B. and C. below. In the remainder of this section, we will outline the overall approach of the investigation.

The approach traditionally used to evaluate the potential of ASR for a particular application has been to conduct what are called [Ref. 7: p. 4] "evaluation experiments". Evaluation experiments are the most rigorous type of experiment and usually involve careful control of experimental conditions, a number of replications, and a formal analysis of numerical measurement data. References 8 and 9 are good examples of the use of evaluation experiments to determine the usefulness of speech recognition in a particular application environment. There are a number of advantages to this approach, the most obvious being that the conclusions are bolstered by hard statistical evidence. The disadvantage is that it is often difficult and expensive to develop experimental models that realistically portray real-world situations.

For a number of reasons, we decided not to use a formal evaluation experiment for the CNCC study. First of all, the CNCC is a complex man-machine system that is extremely difficult to model realistically in a controlled experiment. Events in the network are not deterministic and by introducing the controls necessary for a formal experiment, we run the risk of building a model that does not accurately portray the real system. Related to this is the difficulty of obtaining people with adequate background to serve as experimental subjects. CNCC operators were not available and even subjects with extensive computer experience would have to have lengthy specialized training to participate in such an experiment. The fact that we were looking both at voice input and voice output was an additional complication that could necessitate additional replications and hence increase the cost of the experiment.

A more positive reason for adopting a different strategy was that a formal experiment was really not necessary. Automatic Speech Recognition has matured to the extent that we can feel reasonably confident in extrapolating results from earlier experiments. One of the landmark experiments in ASR involved a group of 24 experimental subjects who used voice recognition equipment to verbally enter commands to the ARPANET [Ref. 8]. The subjects carried out a predefined scenario using both voice input and typing input. The subjects were also required to perform a secondary task during any free time that they had. The results of the experiment [Ref.8: p. 2] showed that with only three hours of training practice:

-- voice input was 17.5% faster than manual typing;

-- manual typing input had 183.2% more entry errors; and

-- voice input allowed subjects to complete 25% more of the secondary task than did manual typing.

It is clear that there are a number of similarities between the scenario in the ARPANET experiment and the operational environment in the CNCC. The CNCC operators are also involved in entering commands to a distributed computer network that uses ARPANET technology; the operator is, of course, always performing a secondary task since the Network

Controller is continually occupied with responding to output requests on a number of display consoles. We believe, then, that the general conclusion of the experiment, that it is feasible to use current commercially available voice recognition equipment to run many operations of an ARPANET type network, can also be applied to the CNCC.

In terms of voice output there is less hard experimental evidence of applicability. There are some examples, however, where voice response units have been used in place of traditional alarm devices; it seems intuitively attractive, furthermore, that voice output would provide considerably more information than a buzzer or bell.

The relatively moderate cost of current voice input and output equipment is also a factor in deciding against a full-scale evaluation experiment. As long as there is reasonable evidence that the technology will be useful, there is fairly low economic risk in trying to apply it. If the basic applicability can be verified; the feasibility demonstrated; and one or more possible implementation alternatives developed, there is a high probability of successful installation and operation.

Our methodology, therefore, has taken two different, but related, paths. First, we have researched the field of VIO technology to understand the capability and cost of the available equipment and its potential applicability in the CNCC. Second, we have implemented a training simulation model of the CNCC that is useful both to demonstrate the use of VIO in the CNCC and to permit experimentation with alternative implementation approaches without interrupting the production system. The model can also be used to familiarize CNCC operators with voice technology equipment before actually installing it on line. Based on these two parallel efforts, we then developed some strategies for installing VIO in the current CNCC without major modifications to any of the existing hardware or software.

B. VOICE INPUT

1. Typical Applications of Voice Input

Until the mid-1970s, automatic speech recognition (ASR) was used primarily to develop vocoders for narrow-band speech communications. With the increasing availability of high quality, moderately priced systems for speech recognition, however, the number of potential and actual applications has mushroomed. Limited vocabulary voice input systems have moved out of the laboratories and are now operating in a variety of applications in both private and public sectors. Examples include automated sorting systems for the distribution of parcels, containers, and baggage; voice programming for machine tools; quality control and inspection; air traffic control; entry of cartographic data; and aids for the handicapped [Ref. 10: pp. 182-186]. While this large number of potential uses precludes detailed discussion of them all, it would be well to briefly discuss a few of them before we try to generalize about the characteristics of applications where ASR is likely to be successful.

Voice input devices have been used in material handling applications since the mid 1970s and have resulted in increased operator productivity and reduced error rates over their predecessors that required operators to hand key the sorting destination codes. A typical example, is the voice control package routing system developed at S.S Kresge in 1974. As parcels arrive at the induction station, the operator simply states the destination code for each package into his microphone headset. The computer recognizes the spoken destination code and generates the appropriate code as if it had come from a keyboard. This "sorting by speech" increased productivity and eliminated a serious problem of "bunching" of different size packages at the induction station [Ref. 10: p. 185].

A longstanding bottleneck in computer-based manufacturing control systems has been the delay in getting machine tool programming requests translated into working software. Traditionally, factory personnel would identify necessary modifications in the form of drawings; these would be converted to specifications; translated into a computer program manually; converted to tape; and, finally, installed in the machine control unit. A form of voice programming has been developed that allows factory supervisors to directly control the computer-based processes. Voice programming for numerical control (VNC) allows the factory personnel to speak commands in stylized English that are automatically translated into a computer-compatible format. No special programming skills are required and the "programmers" hands are free to handle blueprints or perform supporting calculations. VNC has transformed what was a seven step process fraught with the danger of error that creeps in whenever complicated human communications are required into a workable four step process under the direct control of the people using the results [Ref. 10: pp. 185-186].

ASR has not been used as extensively in the public sector although experimental results and limited operational experience seem quite promising. As noted above, Poock [Ref. 8] has demonstrated that using voice input to exercise a typical scenario on the ARPANET was significantly faster and more accurate than manually entering the commands. It has also been used operationally at CINCPACFLT to access a large intelligence data base and has been found to have considerable potential for use in the production of imagery interpretation reports [Ref. 9: p.95]. A significant amount of research has been performed for the Defense Mapping Agency (DMA) on ways to use ASR in such applications as the processing of Digital Landmass System (DLMS) data, preparation of Flight Information Publications data, and ocean-depth measurements for digitized cartographic applications. These operations all take place in an an environment where both hands and eyes are continuously busy and frequently involve the use of stereo optics or other special equipment. Voice has been shown experimentally to be faster, easier, and less fatiguing to the operator than more traditional methods of data entry [Ref. 9: p. 37].

Lest this discussion appear too one-sided, there are many applications for which voice output is either not desirable or not cost-effective. It has been shown to be of limited utility for the preparation of proforma messages and, to be no faster than typing for entering commands to the Warfare Environmental Simulator (WES), a wargame used on the Advanced Command and Control Architectural Testbed (ACCAT). Because of limitations of today's technology, applications that require speaker independence, vocabularies greater than about 300 words, or recognition of continuous speech are not practical. From the amount of experience we have had to date, however, it is possible to identify certain characteristics that make an application a good candidate for voice input. We will discuss these characteristics in the next section.

2. Characteristics of Candidate Applications

In view of the growth in operational use of voice input it should be possible to identify a set of task situation characteristics where the use of speech recognition is likely to be beneficial. Reddy [Ref. 11: p. 60] presents such a list that he adapted from an earlier study. Using this list as a starting point, we can define a profile of characteristics that make a particular application a good candidate for ASR. It is unlikely that many operational situations will exhibit all of these features, but the existence of even two or three provides a strong suggestion that the use of speech input should be examined.

The single characteristic that almost all of the successful applications have in common is that they all involve situations where the operator's hands and eyes are already busy with other functions. In many operations a good deal of productivity is lost when the operator has to interrupt his other activities to input data via a keyboard. Voice input provides an independent, high-bandwidth communication channel that is tailor-made for hands-busy operations.

Related to the first consideration is the condition where the operators must frequently move away from the basic input station to attend to other duties. Wireless transmitters the size of a cigarette pack can provide a considerable range of operation permitting the operator to communicate with an ASR system without returning to the input console. It is interesting to observe that the use of voice input in such situations might suggest, or even necessitate, the use of some type of audible feedback system (i,e. speech output) since the operator will not always be in the vicinity of the speech input device or its associated teletype or CRT.

Some applications involve what might be called "virtual" mobility requirements, in addition to or instead of, the "physical" mobility just described. For example, it may be necessary to access a number of different computer systems through a common terminal or console. The ARPANET experiment provides a good example. In the ARPANET, the host computers use different command languages, query languages, and file management systems. This heterogeneity can sometimes be confusing to even experienced users. For example, the UNIX command to delete a file is 'rm'; while the comparable DECsystem-20 command is 'DEL'. To display a directory of files on UNIX, the user types 'ls -l'; on the DEC-20, he types 'DIR'. Well-designed voice input systems can alleviate these problems, to at least some extent, by making these different command languages transparent to the operator. For example, one widely-used system allows for the definition of vocabulary sets that can provide a mapping between standard input phrases and different output character strings depending on the application context or the particular host computer being accessed. In our second example above, the operator might always utter the phrase "display directory" and the voice input system, in cooperation with the host computer, would perform the transformation to the appropriate character string. Voice input can also reduce complex connection protocols to one or two natural phrases that make it easy to establish logical connections.

Voice input systems are ideal for use by untrained users. No physical skills, such as typing or other keyboard manipulation, are required. The naturalness and humanness of the speech input interface make it applicable for users at all general skill levels: from clerical personnel, to computer operators, to top-level managers.

Operations where speed and accuracy of communication are essential are also excellent candidates for voice input. In the ARPANET experiment, significant improvements (see above) in the speed and accuracy of data entry were attained even though the test subjects had no previous experience with voice input. Some studies have found that peak efficiency is not attained until the operators have had 3-4 months of experience. This suggests that even greater productivity gains can be realized.

Because of technological limitations, which we will pursue in detail in the next section, candidate applications should have a relatively limited input vocabulary. Contemporary voice recognition systems will support vocabularies of from about 50 to over 250 discrete utterances or phrases. Applications with a well-defined but limited command set are the most likely candidates. Most of the vocabulary should be capable of being "canned"; a language that contained many fields with arbitrarily varying input would, therefore, not be very suitable. For a related technological reason, applications with an input language made up of discrete commands and fields is much more adaptable to voice input than one where input is in the form of unstructured character strings. For example, current technology would readily support an information storage and retrieval system using a formatted query language but would be inappropriate for a word-processing application.

Having defined this profile, it will be instructive to compare it against the CNCC operation to see how well the CNCC shapes up as a voice input candidate. Figure 8 lists the six characteristics and a subjective assessment of how well each applies to the CNCC. We see that, with two exceptions, all the features are present in the Network Control Center. The Network Controller is often busy doing something else when he is called upon to input commands to the system; the need to control a number of systems clearly requires physical mobility; the responsibility for maintaining an operating CCINS network requires that accurate information be entered quickly; and the CNCC command language is small and reasonably well structured. The CNCC personnel are adept at entering commands and data

CHARACTERISTICS	APPLY IN CNCC?
1. Hands busy, eyes busy	Yes
2. Physical mobility	Yes
3. "Virtual" mobility	Possibly
4. Untrained users	N/A
5. Speed and accuracy required	Yes
6. Small, structured vocabulary	Yes

Figure 8. The CNCC as a Candidate for Voice Input

into computer systems so the benefits to untrained users are not really relevant.

The only uncertain area is that which we have called "virtual" mobility. There is no question that such a requirement exists in the CNCC. Given the number of systems for which the Network Controller is responsible even a limited amount of connection-protocol simplification and command language transparency would be quite beneficial. The difficulty is that voice input can only help if the systems in question are already connected electrically and they have an agreed-upon protocol for connection, even a basic timesharing protocol such as ARPA's TELNET. It does not appear that such electrical connections and protocols exist today, e.g. there appears to be no method of accessing the other machines in the CNCC complex directly from the Summary TTY. Nonetheless, in view of the overall assessment shown in Figure 8, the expectation is that voice input can be of considerable benefit to CNCC operations.

3. Overview of Voice Input Technology

Automatic Speech Recognition is a subset of a broader intellectual domain known as Speech Understanding. Speech Understanding systems (SUSs) have the objective of properly interpreting the intent of a speaker even when the speech is not grammatically correct or well-formed. SUSs must, for example, take into account the tendencies of speakers to talk in incomplete context-sensitive sentences and to sprinkle utterances with occasional non-speech elements such as "uhs", "Ya knows", etc. It is important to note that SUS are not so much interested in correct recognition of every word but rather focus on the meaning of entire conversational segments.

Even when the domain is relatively restricted, permitting the use of task-specific information, the job of a speech understanding system is truly formidable and efforts to date have been mostly of academic interest. In the speaking process, a speaker transforms concepts into speech through processes that introduce variability and noise. In an attempt to understand the concepts, the system must deal with phonemic, lexical, syntactic, and semantic levels of meaning all of which are susceptible to the introduction of additional noise or error. In the words of the designers of one of the prototype SUS,

> To comprehend an utterance in the context of such errors, a speech understanding system must formulate and evaluate numerous candidate interpretations of speech fragments. Understanding a message requires us to isolate and recognize its individual words and parse their syntactic and conceptual relationships [Ref. 12: p. 216].

The goals of Speech Recognition, in contrast, are much less ambitious. Instead of dealing with abstract concepts like meaning and understanding, speech recognition systems attempt to solve the more practical problems of analyzing the acoustic waveform and applying pattern recognition techniques to differentiate between utterances.

ASR systems can be considered as belonging to one of two categories: continuous (connected) speech systems or isolated (discrete) speech systems. (Some authors draw a distinction between continuous and connected speech, regarding the latter as a somewhat simpler problem. In this thesis the terms are used interchangably.) While the distinctions between the categories are often blurred, discrete systems are those that require a short pause (on the order of 100 to 200 ms) between utterances; Connected systems, on the other hand, can recognize words without explicit knowledge of their endpoints. Continuous speech recognition is considerably more difficult than recognition of discrete utterances. First of all, such systems must decompose the acoustic waveform into phonemes and words at a real time rate. This naturally requires a great deal of computational power. Secondly, the acoustic variation of words spoken in connected speech is different than when the words are spoken discretely. This is caused by the coarticulation of neighboring sounds, i.e. the positions of the tongue, jaw, and lips in a speech sound are affected by the previous and future positions. Continuous recognizers must, therefore, be very sensitive to conversational context [Ref. 13: p. 27]. There are some connected-speech recognizers on the market today but they are expensive

($50,000-$100,000) and their capabilities have not really been evaluated. For the remainder of this thesis, then, we will confine our discussions to discrete recognition systems which are commercially available in price ranges from as low as $500 upwards to about $15,000.

There are two other limitations with most of the contemporary systems. The first deals with limitations on vocabulary size; the second concerns speaker-independence. Commercially available systems support vocabularies of about 40-256 discrete utterances or "words". (In speech recognition parlance the terms "word" and "utterance" are used interchangeably, even though the term "phrase" might be more appropriate since utterances are often composed of more than one word. In addition most systems are speaker-dependent, i.e. they require a user to first train the system with his voice patterns for each of the utterances in the vocabulary before using the system. (An exception is the type of recognizer that supports only a very specialized vocabulary, e.g. the 10 digits). For most applications, neither restriction has much practical effect. Very few computer-related applications require more than 200 utterances and training is usually accomplished quickly and easily.

Figure 9 shows a block diagram of a typical discrete word recognition system. While the diagram and the discussion to follow describe a particular implementation

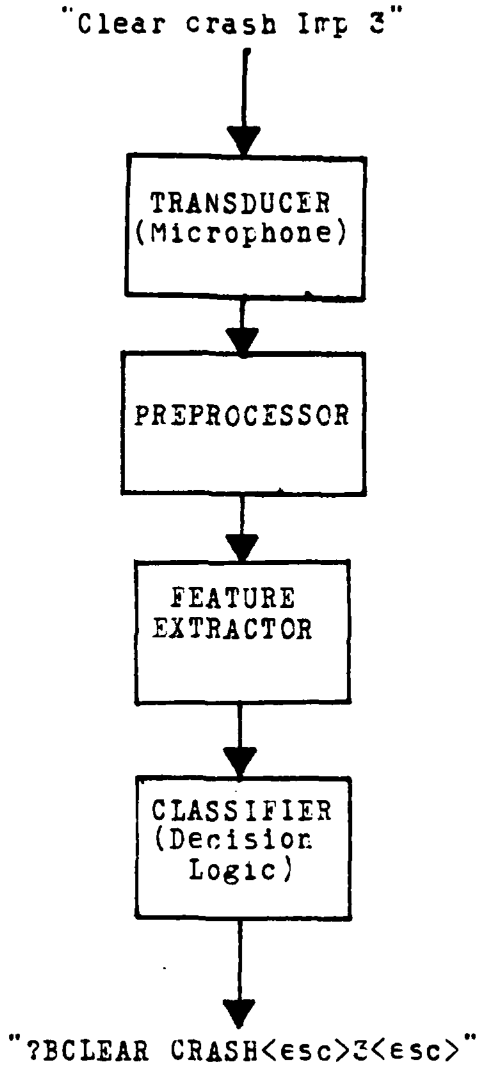


Figure 9. Block Diagram of Typical Speech Recognition System (From Reference 10)

described by Thomas B. Martin of Threshold Technology, Inc. [Ref. 10: pp. 176-178], the general approach applies to most of the systems available today. As the diagram shows, the system takes a spoken utterance, attempts to match it with one of its pre-stored voice patterns, and, if successful, outputs a pre-defined string of characters over a standard RS-232 interface to a host computer. In our example, the spoken phrase "Clear Crash Imp 3" generates the output character string "?BCLEAR CRASH<esc>3<esc>" which is the proper operator entry at the Summary TTY in response to a number of observed network problems (e.g. IMP CRASH messages, PACKAGE = n messages, etc.) This example illustrates a number of important points about ASR. First, the spoken utterance is considerably more natural than the typed command string. Second and probably more important, the voice input device is completely transparent to the host computer (in this case the CNCC machine). As far as the host is concerned, it is getting character strings in the same code and at the same bit rate as if they had been typed manually. Thus no host software modifications are required in the host. This is critical in light of the complexity of the CNCC program, the limited primary memory, and the consequent difficulties in making software changes.

We see from Figure 9 that a system consists of four basic components: a microphone transducer, a preprocessor, a feature extractor, and a final decision level classifier.

Early speech recognition systems were weakest in terms of feature extraction, often deleting the function completely or using a rudimentary form of template matching which proved inadequate and was soon rejected.

The microphone, of course, is the interface between the user and the system and converts the spoken phrase into an electrical signal that can be analyzed by the other components. The preprocessor performs a number of functions on the electrical signal. It normalizes the signal in time so that it can be later compared with the reference patterns. Dynamic programming is one technique used to achieve this time adjustment or warping. The preprocessor also performs classical time or frequency domain analysis on the input signal. There are two approaches commonly seen: the first uses bandpass filtering and Fourier analysis in the frequency domain; the second uses Linear Predictive Coding (LPC) to analyze the signal in the time domain.

Feature extraction is the most important processing function in any pattern recognition system of which speech recognition systems form a major subset. Both the spectral shape and the time derivative of the spectral envelope are measured over the frequency range of interest. Combinations and sequences of these measurements are used to produce a set of 32 acoustic features which include such things as a word boundary estimate, spectral amplitudes, and formant frequencies.

In most of today's systems the first three functional components have been implemented exclusively or primarily in hardware in order to achieve real time processing. In contrast, the classification or decision process is usually implemented in software on a mini or microcomputer. The classifier uses pattern-matching logic to compare the features of the utterance with the pre-stored reference patterns and uses a best-fit algorithm to determine the correct one. It is also possible to produce a "no-decision" or reject when none of the reference patterns is close enough to the utterance. When the classifier makes its decision, it uses the number of the best-fit utterance to select the correct output character string which it then sends on a serial interface to a host computer.

There are two types of errors that can occur in speech recognition. The first is rejection or the inability to correctly classify an utterance. The second, and more troublesome, occur when the recognizer substitutes one utterance for another. Rejection errors typically cause few problems since most recognizers will inform the user by an audible beep that an utterance was rejected. Substitutions are more difficult and the better systems usually have recognition algorithms designed to reject rather than guess at questionable words. The better quality systems, such as the Threshold Technology 600 and 680, have error rates that are quite acceptable. In the imagery interpretation experiment, for example, recognition accuracy was 97% if only substitutions were counted as errors. Even if rejects were included in the error counts, recognition was still better than 95%. These figures compare quite favorably with results obtained from manual typing. Results of other evaluations have been roughly the same. We can say with some certainty, then, that limited vocabulary, speaker-dependent voice input systems now represent stable and mature technology that can be used in a number of applications, and certainly in the CNCC.

C. VOICE OUTPUT

1. Typical Applications of Voice Output

In contrast to voice input technology, the use of voice output is not so well-defined either in terms of the technology itself or of its applications. Manufacturers of voice response equipment have yet to settle on a common technological approach and application outside the telephone industry has been comparatively limited. In general, there are three broad areas where the use of speech output devices has proven beneficial. Before discussing the technology further, it is worthwhile to look at these three areas more closely.

The first application area involves those operations where the telephone has always been an integral part of day-to-day operations. The most obvious example is in the phone industry itself where computerized speech has been used for some time to perform functions that have traditionally been done by human operators (e.g. "The number you have dialed is not in service", etc.) Voice response has also been used in a number of private automated switching systems such as MCI EXECUNET, a product of MCI Telecommunications Corporation which links a number of Cognitronics 681 Speechmakers to the Universal Switched Network of Danray, Inc., a division of Northern Telecom [Ref. 14]. Another type of telephone application has been the implementation of phone order-entry systems. The Ford Motor Company's Direct Order Entry System (DOES), for example, has been in operation since the mid-1970s and has been very well-received by Ford and Lincoln-Mercury dealers. In this application, the computerized speech output prompts the dealer through the parts ordering process. Major benefits have been "instant stock status, greater accuracy, improved stock turn, and ... overall improved customer service." [Ref. 15]

The second broad application area is the use of computerized speech to provide information traditionally supplied by a human speaker. An example is an experimental system developed for the National Aeronautics and Space

Administration (NASA) designed to test the concept of automatic approach callouts. This system (called SYNCALL) used synthesized speech to generate the aperiodic voice reports given a commercial airline pilot by others in the cockpit crew during the approach and landing phases of flight [Ref. 16: pp. 4-9]. Although SYNCALL was developed as an experimental system, it did result in improvements in flight performance and was generally favored by pilots over the traditional callouts by the cockpit crew. Improvements were most marked for approaches with high manual and visual workload [Ref. 16: p. 78]. Variations of this idea have used speech to replace outputs usually generated by analog or digital readout devices, such as altimeters and fuel gauges.

The third major application area of voice output is as a replacement for traditional alarm devices such as lights, bells, or sirens. The principal advantage of speech output in this context is that it conveys considerably more information than the other alarm types. If, for example, a computer operator hears a bell sound on the computer console, he first must go to the console and read the typed output before identifying the problem and deciding on corrective action. On the other hand, if he hears the phrase "printer number two is jammed", he can immediately go to the device in question and correct the problem. It is, of course, this application area that we are most interested in for the CNCC. As we discussed in Section III, the CNCC Controller is very busy and quite often away from the Logger teletype. Replacing, or supplementing, the more important Logger TTY messages with voice response might permit him to do his job faster and more effectively and save steps in the process.

In our discussion of speech recognition, we briefly mentioned voice output as a feedback mechanism in a voice input system. This might be considered an auxiliary application of speech output since its fundamental raison d'etre is to close the interactive loop with the voice input user. An operator may be some distance from the speech recognizer that is processing his utterances. It is inconvenient to have him walk to a TTY to receive feedback on his input commands. Voice response is a logical candidate for the feedback mechanism. In the current thesis we are focusing primarily on the alerting capability of speech output although there is limited use of aural feedback in the CNCC model. The CNCC computer provides only limited feedback as it stands today, making the design of a total closed-loop voice input/output system impractical. This subject deserves a good deal more study and would seem to be a good approach to use in the CNCC replacement system. The technology is clearly here; it remains only to apply the technology to the design of a voice-controlled man-machine communication interface.

2. Overview of Voice Output Technology

While development of speech recognition systems was not possible before the introduction of inexpensive electronic computing equipment, "artificial speech" systems have much deeper historical roots. As early as the Renaissance, man began to look for ways to simulate the actions of the complex vocal mechanism by mechanical contrivances, many of which were quite clever in design [Ref. 17: p.205]. Fairly elaborate speaking machines were constructed by scientific pioneers like Kratzenstein, Von Kemplen, and Sir Charles Wheatstone (no doubt more famous for the Wheatstone Bridge). These early mechanical efforts usually involved the use of bellows that generated varying air currents through a vibrating reed.

As a boy, Alexander Graham Bell and his brother built a speaking automaton by making a cast from a human skull and molding the vocal parts in gutta-percha. One can only speculate on the part that this youthful endeavor played in Bell's later work on the telephone. As described by Flanagan [Ref. 17: pp. 206-207],

> The lips, tongue, palate, teeth, pharynx, and velum were represented. The lips were a framework of wire covered with rubber which had been stuffed with cotton bailing. Rubber cheeks enclosed the mouth cavity, and the tongue was simulated by a rubber skin and stuffed with batting. The parts were activated by levers controlled from a keyboard. A larynx 'box' was constructed of tin and had a flexible tube for a windpipe. A vocal cord orifice was made by stretching a slotted rubber sheet over tin supports.

All in all it was apparently quite a creation that could utter vowels, nasals, and a few simple phrases.

Interest in mechanical analogs of the human vocal system continued into the twentieth century. Later approaches tended to depart from the human physical representation and turned instead to devices such as tuning forks and organ pipes. The evolution of electrical technology accelerated the move away from physical models in the direction of systems that generated sounds by electrical tuning of amplitude and frequency. H. W. Dudley demonstrated his Vocoder or "talking machine" at the 1939 World's Fair. In the 1950s and 60s, accustic engineers made considerable improvements in analyzing and representing the human vocal tract. Advances in computer technology and the sharp reductions in cost of electronic components have advanced the state of the art to the point where reasonable quality synthetic speech is available at relatively moderate cost.

The basic functional components of a computer voice response system are shown in Figure 10. The machine is required to speak a phrase typically expressed in spoken or typed English text. The synthesis program must somehow translate the original text into a digital stream that represents the desired output including, if possible, the proper duration, intensity, and inflection for the prescribed context. This stream is then input to a digital

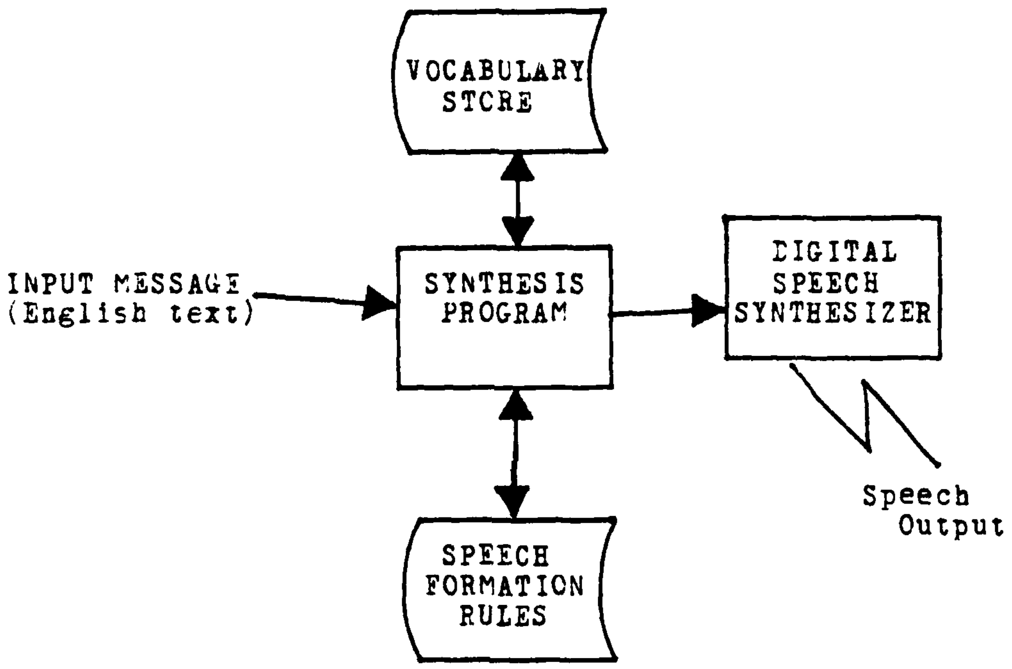


Figure 10. Block Diagram of Computer Voice Response System (From Reference 17)

speech synthesizer which "speaks" the message either over a telephone or through a local speaker. While the functional boxes are similar for all voice response systems, there is considerable difference in the way that the synthetic speech is produced.

There are three basic approaches to speech synthesis; they are distinguished largely by the storage capacity needed for the vocabulary and by the complexity of the control rules for generating the speech. These techniques are: adaptive differential pulse-code modulation (ADPCM), formant synthesis, and text (or phoneme) synthesis [Ref. 17: pp. 5-6]. Their respective data rates and bandwidth requirements are compared in Figure 11.

ADPCM is the simplest technique. It uses a vocabulary of human-spoken words whose waveforms are digitally coded. The method requires tradeoffs between signal quality, storage capacity required for the vocabulary, and the simplicity of the message-generating program. For message assembly, the synthesizer retrieves the digitally coded words from a disk storage device and applies them to an ADPCM decoder which produces the analog output signal. With this technique there is no way to control prosody; that is, no control of vocal pitch or merging of vocal resonances is possible. For this reason, the most appropriate applications are those with minimal possibility for semantic ambiguity. ADPCM has been used to

CODING	DATA RATE	AMT OF STORED SPEECH IN 1 MILION BITS
ADPCM	20K bits/s	1 minute
FORMANT SYNTHESIS	500 bits/s	32 minutes
PHONEME SYNTHESIS	75 bits/s	240 minutes

Figure 11. Data Rates for Different Synthesis Techniques (from Reference 17)

generate voice output in applications such as equipment assembly, numeric readout, and stock market quotations[Ref. 17: p. 6].

The second voice response technique, formant synthesis, uses a method in which a word library (again initially spoken by a human) is analyzed and stored as the time variations of vocal-tract resonances or formants. These frequency variations are computer analyzed and used to drive a digital filter whose input excitation is derived from programmed rules for voice pitch and sound amplitudes. Formant synthesis has been used for the same application areas as ADPCM.

The third, and in some ways the most advanced, voice response technique is called text synthesis or, alternatively, phoneme synthesis. It generates speech from a data input operating at a typewriter rate, i.e. about 75 bits/sec. Systems that use this technique generate the voice output entirely from stored rules and dictionaries. Phoneme synthesis, thus, makes possible the direct conversion of typed English text to synthetic speech. This flexibility does not come without cost. Text synthesis systems are invariably software-intensive. Routines are required to convert the text strings to phonemes and apply prosodic rules to make the speech sound "normal". Such software is quite complex and usually can only be written by someone who combines software development skills with formal training in Linguistics, a combination not frequently found. In spite of this imposing software requirement, phoneme synthesis is probably the best for computer applications since it provides maximum flexibility and has a data rate that is compatible with standard computer peripherals. The device used to run the CNCC model, the VOTRAX ML-I, uses the phoneme synthesis approach. The field is quite volatile, however, and new and less expensive products are being introduced all the time. Improvements in digital recording and compression techniques and comparable advances in the parameterized waveform (formant synthesis) method may mean that text synthesis does not represent the best long-term alternative [Ref. 18: p. 74].

Voice output is available in three forms: chips, boards, and terminals. Chips and boards are inexpensive and self-contained (i.e. the speech waveforms are locally stored) thus they don't normally rely on a host mainframe. They are fairly inflexible, however, and require considerable expertise to integrate into a workable system. Prices for voice output terminals start around $500 and go as high as $50,000; most are priced under $10,000. These terminals are designed to connect directly to computers via an RS-232 or 20mA serial loop interface and usually rely on external memory for vocabulary storage. Most of the terminals use phoneme synthesis to generate the output speech. Because of their ease of use and greater overall capabilities, voice terminals would seem to be more appropriate for the CNCC application than chips or boards.

V. THE CNCC MODEL

A. DESIGN GOALS AND CONSIDERATIONS

The CNCC model is a discrete event simulation that models the external behavior of the COINS Network Control Center computer. It does not attempt to model the internal operations of the CNCC, nor to simulate the flow of messages and packets across the COINS II network. Instead, it focuses on modeling the system from the point-of-view of the Network Controller, i.e. it simulates the output behavior of the Logger TTY and the input and output capabilities of the Summary TTY. Design and implementation were driven by the following six design goals.

1. The model should serve as a realistic simulation of the external behavior of the CNCC computer and operating system.
2. It should provide a vehicle to demonstrate the feasitility of using voice input and output technology as an integral part of the system.
3. It should be able to serve as a voice technology experimental testbed. It must be able to run in several different operating modes and to measure variations in user response times and overall performance under these different operating environments.
4. The model must be implemented on a computer system that is accessible to the COINS PMO.
5. It should be modularly designed and independent, as far as possible, of any specific voice input or output hardware.

6. The model should be extensible and easy to modify so it can serve as a long term experimental testbed for the CNCC.

The most important of these objectives is that the model be a realistic portrayal of the CNCC environment. It is essential that the model behave in a manner consistent with the way that the real CNCC computer acts. For example, the model must drive two independent terminal devices, one for the Logger TTY and one for the Summary TTY. Processing of operator commands should be identical to the way that the CNCC handles command input, i.e. the responses and prompts must be the same as on the real CNCC. A fairly complete subset of the CNCC command language must be supported, to allow the operator to perform the Network Controller functions through the model. Network events should occur in the model in the same way that they do in real life. Failures and errors in network components are, of course, not deterministic. The model, therefore, should use stochastic methods to generate the network events. If patterns of events were predictable, it would be impossible to discount the effects of learning when running the model. It would be considerably more difficult to evaluate response time changes under the various operating modes.

Since the model's main purpose is to explore the feasibility of voice technology, it must be capable of accepting input from a speech recognizer and sending output to a speech synthesizer. Speech input is, as noted in the previous section, transparent to the host computer since the recognizer appears like a terminal device on an RS-232 serial interface; voice output, on the other hand, is not nearly so transparent. One of the key software routines in the model is a generalized phoneme output procedure that interfaces with the VOTRAX ML-I synthesizer. The routine is table-driven so it should be reasonably easy to modify it for other phoneme synthesizers. In addition to the voice output routine, we designed and implemented a general-purpose text-to-speech program that allows a user to create, store, and modify phoneme strings from directly from English text input.

One possible use of the model is to try to quantify improvements attributed to the use of voice input or output. To do this, it should have the ability to measure user response times in at least three modes: manually typed input and printed Logger TTY output; voice input and printed Logger output; and voice input and output. By examining the results of these response time measurements, it may be possible to make an assessment of response time as a function of input or output mode.

To have real long term value, the model should be able to be used by COINS PMO and CNCC personnel to investigate voice technology design tradeoffs. For example, we may want to look at the effects of different input vocabularies or alternate phrasing of output messages. The model should provide the framework for studies of this nature. For this reason, the model was implemented on a PDP-11/70 computer running the UNIX operating system. It should be easy to transport to one of the COINS Terminal Access Systems.

Modular design and device independence carry with them the distinct connotations of motherhood and apple pie. Nevertheless, we did make a real effort to embody these features in the CNCC model. The program is made up of a number of small, single-function routines and, except for the VOTRAX-dependent code, avoids ties to specific physical devices. The UNIX I/O system, which views all devices as files, helps maintain this generality. To interface to another voice output device would require only that a new output driver be written. The message definition and phoneme retrieval logic can remain intact.

The design of the model should permit extensions and modifications to be made without changes to the basic software structure. To ensure this, most of the important procedures are file or table-driven. Parameters and physical device assignments can be changed at run time. Adding new commands or events requires only adding an entry to the appropriate table and writing a handler for the new event. The entire program is written in DEC FORTRAN-IV PLUS as modified for UNIX by CULC Inc. This combines the advantages of writing in a commonly-known language with the ability to use many of the basic features of UNIX.

B. HARDWARE AND SOFTWARE ENVIRONMENT

The CNCC model operates on a Digital Equipment Corporation (DEC) PDP-11/70 computer running the UNIX operating system. The system on which the model was developed, in the NPS C3 laboratory, uses a version of UNIX supported by Bolt Beranek and Newman (BBN). BBN UNIX is a hybrid of the sixth and seventh editions of the standard Western Electric UNIX The model consists of two cooperating processes both coded in CULC FORTRAN-IV PLUS (F4P). F4P is a superset of ANSI standard FORTRAN that provides access to the UNIX I/O system and to many of the standard system calls [Ref. 20]. Since portability to non-UNIX computer systems was not an explicit design goal, we felt free to use special F4P features like byte variables and UNIX system calls. The model should be easy to install on systems running variants of UNIX other than the BBN version.

In addition to the PDP-11/70, the hardware suite consists of two terminals connected to the mainframe via serial RS-232C interfaces. In the NPS configuration, we used two ADM3 CRT terminals manufactured by Lear Siegler Inc. Any terminal that looks like a teletype to UNIX, either hardcopy or CRT, would work just as well. Associated with the Summary TTY is a Threshold Technology T-600 Speech Recognizer used for input of voice commands. When operating in voice input mode, the operator speaks short command phrases to the T-600 which converts them to strings of ASCII characters acceptable to the model's command interpreter. A list of the operator utterances along with the associated character strings is given in APPENDIX C. The line to the Logger TTY terminates in a VOTRAX ML-I Audio Response System. This device normally passes on all characters to an ADM3 connected to its business equipment port. When the ML-I detects special control characters in the data stream, it interprets the characters that follow as a series of phoneme codes making up a voice output message. The voice ouput messages used by the model are summarized in APPENDIX D. A block diagram of the hardware configuration is shown in Figure 12.

The model's software comprises two F4P programs, named cncc and ttyin. In addition, a voice output editor (mil) is used to generate the phoneme strings for the synthesizer. When all programs are considered, the model involves approximately 20,000 executable FORTRAN source language statements. Code and supporting files occupy over 1600 blocks of RP06 disk storage (about 800K bytes). The source code for all software is the property of the U.S. government. Anyone interested in possible use of the programs should contact the author. Operating instructions for the program can be found in the file cncc.hlp. This file is include as APPENDIX A to this thesis.

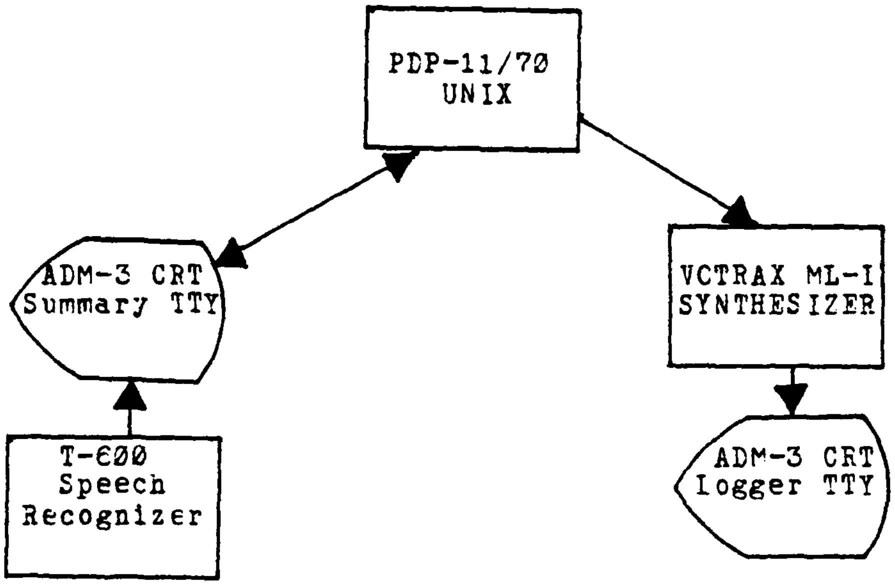


Figure 12. Block Diagram of Hardware Suite for CNCC Model

C. FUNCTIONAL DESIGN

Figure 12 can serve as a useful starting point for discussing the functional design of the CNCC model. After initializing its internal tables and parameters, the model waits for either an internal event to occur or a user command to be entered on the Summary TTY. As events occur, they normally generate messages on the Logger TTY. Certain messages are accompanied by an audible beep that serves as an operator alarm. If voice output mode is enabled these events will produce a voice output message in addition to the printed output. Some events, typically automatic summary reports, are printed on the Summary TTY. The Network Controller can also influence events by typing commands (or speaking them) at the Summary TTY. These commands are often in response to previous Logger messages (for example, the command to reload an IMP following an IMP down or IMP trap Logger message). Figure 13 lists the internally generated events supported by the model. The Network Controller commands that are supported in the initial release of the model are shown in Figure 14.

The list shown in Figure 13 accounts for the most frequently occurring network events. For simplicity, the program assumes that all component failures are exponentially distributed according to some parameter value for Mean Time Between Failures (MTBF). In the runs of the model conducted at NPS, we used a MTBF for both IMPs and

EVENT CODE	DESCRIPTION	RESP REC
11	IMP xx DOWN	Y
12	LINE xx DOWN	Y
13	VIRT-HOST xx DOWN	N
14	BAD HOST DATA CKSUM	Y
15	IMP xx TRAP	Y
16	IMP xx CRASH	Y
17	LINE xx ERRORS (+/-)	N
18	LINE xx DOWN (+/-)	Y
99	REPEAT PENDING ACTIONS	Y

Figure 13. CNCC Network Events

CMD CODE	CMD NAME	ACTION	OUTPUT TTY
1	BROADCAST FILE	3	Log
2	PRINT (Dir or Total)	1	Summ
3	SEND FILE TO HOST	3	Summ
4	DUMP IMP PGM TO DISK	3	Summ
5	DELETE FILE	2	Summ
6	RENAME FILE	2	Summ
7	RELOAD IMP	3	Log
8	SUMMARY REPORT	1	Summ
9	STOP PROGRAM	1	Summ
10	LINE STATUS CHANGE	2	Log

Figure 14. CNCC Operator Commands

lines of one day (86,400 seconds). We also assumed that some types of failures are more likely than others. Error conditions on a line, for instance, are much more probable than the line going down. The program uses a probability distribution that accounts for the relative likelihood of different events along with exponentially distributed random numbers to select and schedule the next event for each component. The parameter for the exponential distribution can be modified by the user, at run time, without changing any program code; the relative likelihood distribution is stored in an internal table, however, and can be changed only by a program recompilation and link edit.

As can be seen from Figure 14, the model does not support all of the CNCC commands. Those most commonly used to respond to network failures, however, are all included. Two of the commands are not part of the real CNCC command repertoire, but were implemented for user convenience. The QUIT command allows the user to gracefully halt the model; the LINE STATUS change command allows the user to set the status of a given line to UP, DOWN, or LOOPED. The command interpreter behaves exactly like the real CNCC. Each command must be preceded by a question mark and the operator has only to type the characters needed to uniquely identify the command. In response to each command, the system will either perform or simulate some action and send an appropriate response to either the Summary or the Logger

TTY. The program will take one of three possible actions: (1) actually perform some service, such as printing a directory of files; (2) simulate an action and type an immediate response, such as simulating a file deletion; or (3) use some probability distribution, such as a normal distribution, to determine the completion time for an action that involves some delay caused by disk or communication transfers. For example, the response to the RELOAD command will be scheduled using a normal distribution with mean 12 and standard deviation 2 seconds.

D. INTERNAL DESIGN

1. Overall Structure

The CNCC model is composed of two cooperating processes that communicate over a specially designed Interprocess Communication Facility (IPCF). An overview of the model's process structure is depicted in Figure 15. The child process (ttyin) performs the single function of accepting and validating commands entered by the operator at the Summary TTY. When a legal command is recognized, ttyin passes a message to its parent process (cncc) that contains all information necessary to execute (or simulate) the command. All functions other than command input are performed by the parent. These include scheduling and executing network events; processing operator commands accepted by ttyin; printing summary reports and other output

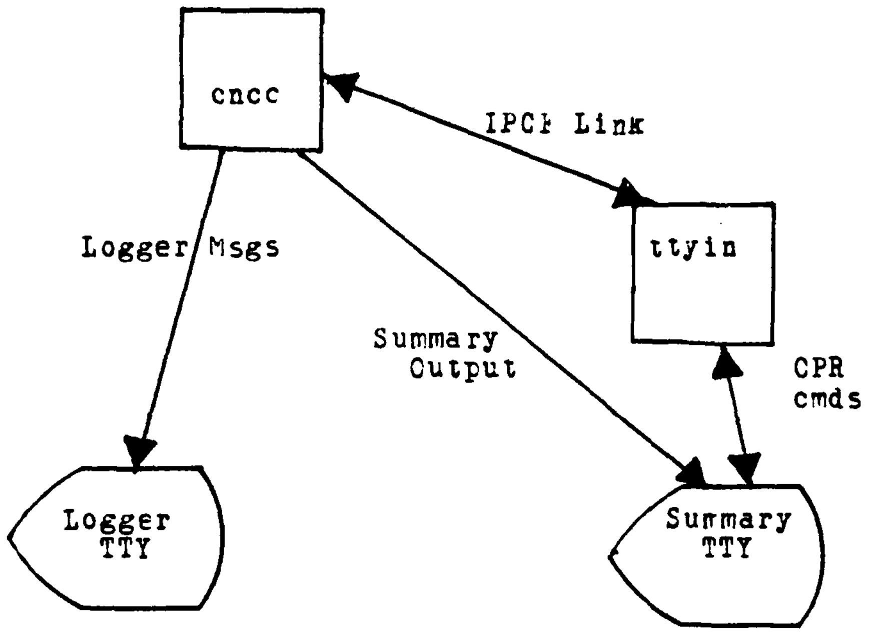


Figure 15. CNCC Top-Level Process Structure

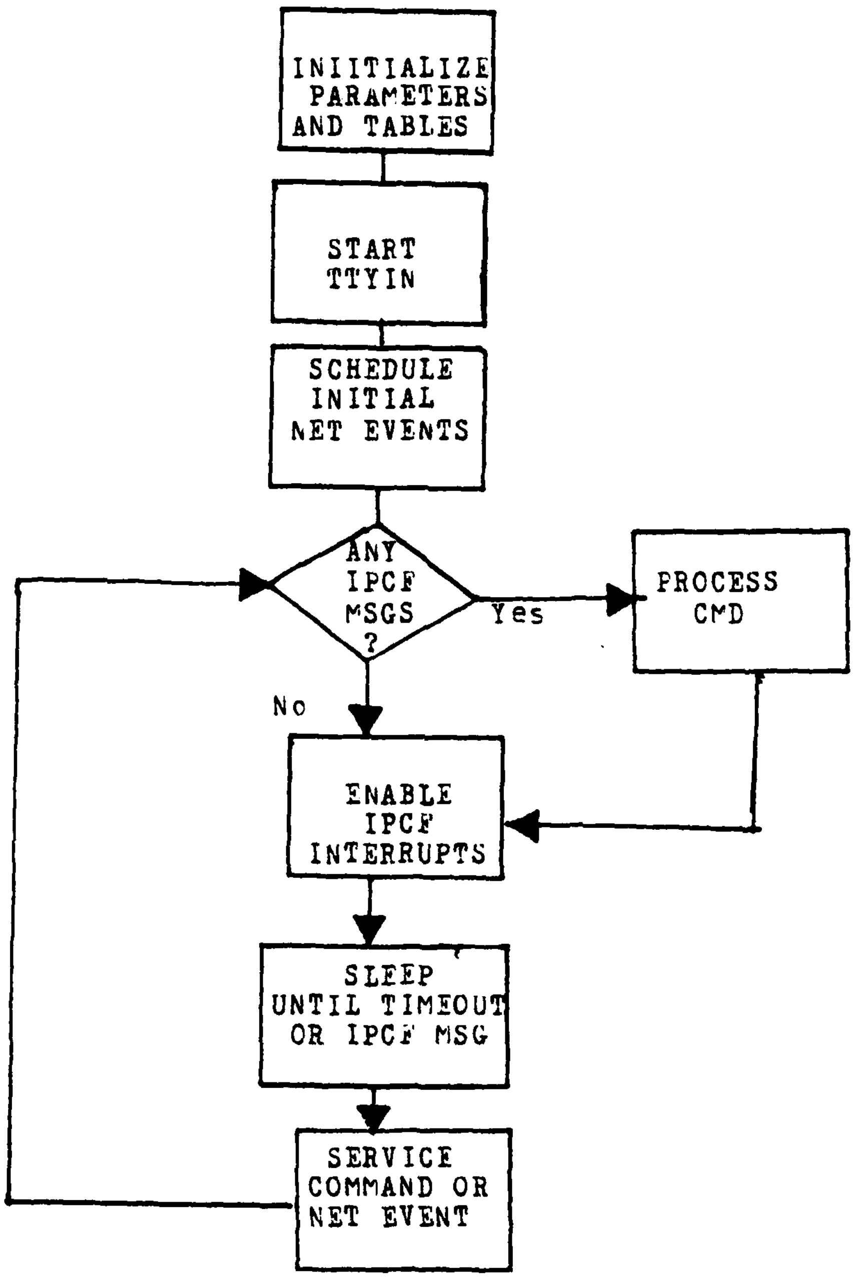
on the Summary TTY; and sending alert messages to the Logger TTY, including voice output messages if speech output mode is enabled.

Before discussing the control structures of the two processes, it will be useful to describe the operation of the IPCF facility. IPCF was needed because the model could not be implemented in a single process. UNIX provides a "sleep" system call [Ref. 20] that allows a program to sleep for some time interval, i.e. until some network event is to be executed. The only condition that will wake the program from a sleep is the passage of the time interval or a special type of software interrupt known as a "signal". A program cannot simultaneously sleep and accept TTY input; conversely, a program in TTY input state can't be interrupted for a time-out. This situation necessitates that command input and event execution be handled separately by different UNIX processes. The processes, although operating asynchronously, do need some way to exchange information. The typical way for processes to communicate in UNIX is by use of the "pipe" mechanism. Use of a pipe requires synchronization between the cooperating processes, i.e. one process has to be ready to read what the other writes on the pipe. A pipe, therefore, doesn't really solve the original problem; it only changes its form. What we did was implement a primitive form of interprocess communication that allows the processes to run asynchronously, but still exchange information.

The approach that we adopted was to use disk files as mailboxes written by the sending process and read by the destination. We used disk files because the UNIX Large Core Buffer Area (LCBA) facility which allows processes to access common main memory was not implemented on the NPS UNIX system. The system was designed strictly for use by the CNCC model and no attempt was made to introduce much generality. IPCF is made up of three FORTRAN subroutines: SNDMSG, RDMSG, and CHKMSG. SNDMSG writes the message to a disk file named XPCF###, where ### is the Process Identifier (PID) of the destination process and then sends a signal to the destination process. Reception of this signal causes the destination process to interrupt its current activity (usually sleep) and execute RDMSG which reads the message from the disk file. CHKMSG is a routine that checks for and reads any messages that might have arrived while the destination process had disabled IPCF interrupts, typically when performing non-interruptable activities such as I/O. IPCF provides double-buffering and is, thus, full-duplex; in practice, however, it is typically used to pass messages in only one direction.

The basic control structure of the parent process (cncc) is shown in Figure 16. The program is event or command-driven, i.e. it runs only in response to network events or operator commands. EVCHK and EVCMD both call a single dispatching routine (PERFRM) which calls the appropriate handler for each event. Events that require operator action, e.g. IMP DOWN, are executed like other events but after execution go on a special table called the Pending Action List where they remain until the correct operator response is recognized.

The control structure of ttyin (see Figure 17) is even simpler. The program's sole function is to accept valid commands from the Summary TTY and send an IPCF message to cncc for each command that it accepts. The only thing that complicates ttyin is the requirement that input be unbuffered, i.e. character-by-character rather than a line at a time. In normal UNIX TTY I/O, the system buffers a line at a time and performs certain local editing functions such as recognizing special characters as delete-character and delete-line flags. To accurately simulate the Summary TTY interface, ttyin must examine each character as it is typed and, consequently, must perform its own local editing (see Figure 18). To do this, ttyin must operate in what, in UNIX parlance, is called RAW mode, where each character is passed as typed to the application program. The handler interprets ctrl-h as the rubout character and ctrl-u as the delete-line flag. When processing the arguments of a command, the handlers recognize the sentinels and escape characters as defined in Reference 3.



INIITIALIZE PARAMETERS AND TABLES
START TTYIN
SCHEDULE INITIAL NET EVENTS
ANY IPCF MSGS ?
Yes
PROCESS CMD
No
ENABLE IPCF INTERRUPTS
SLEEP UNTIL TIMEOUT OR IPCF MSG
SERVICE COMMAND OR NET EVENT

Figure 16. CNCC Process Control Structure

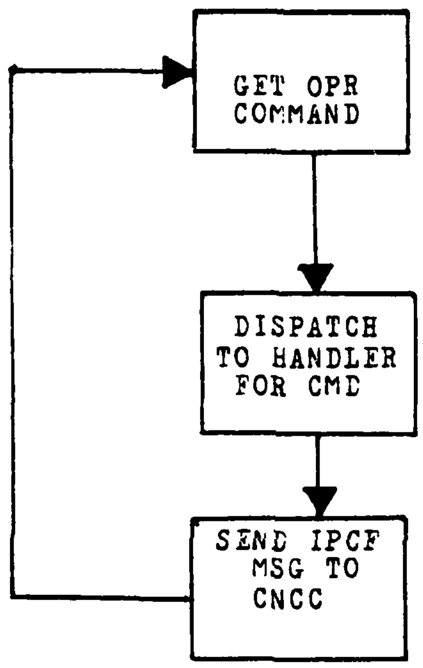


Figure 17. TTYIN Process Control Structure

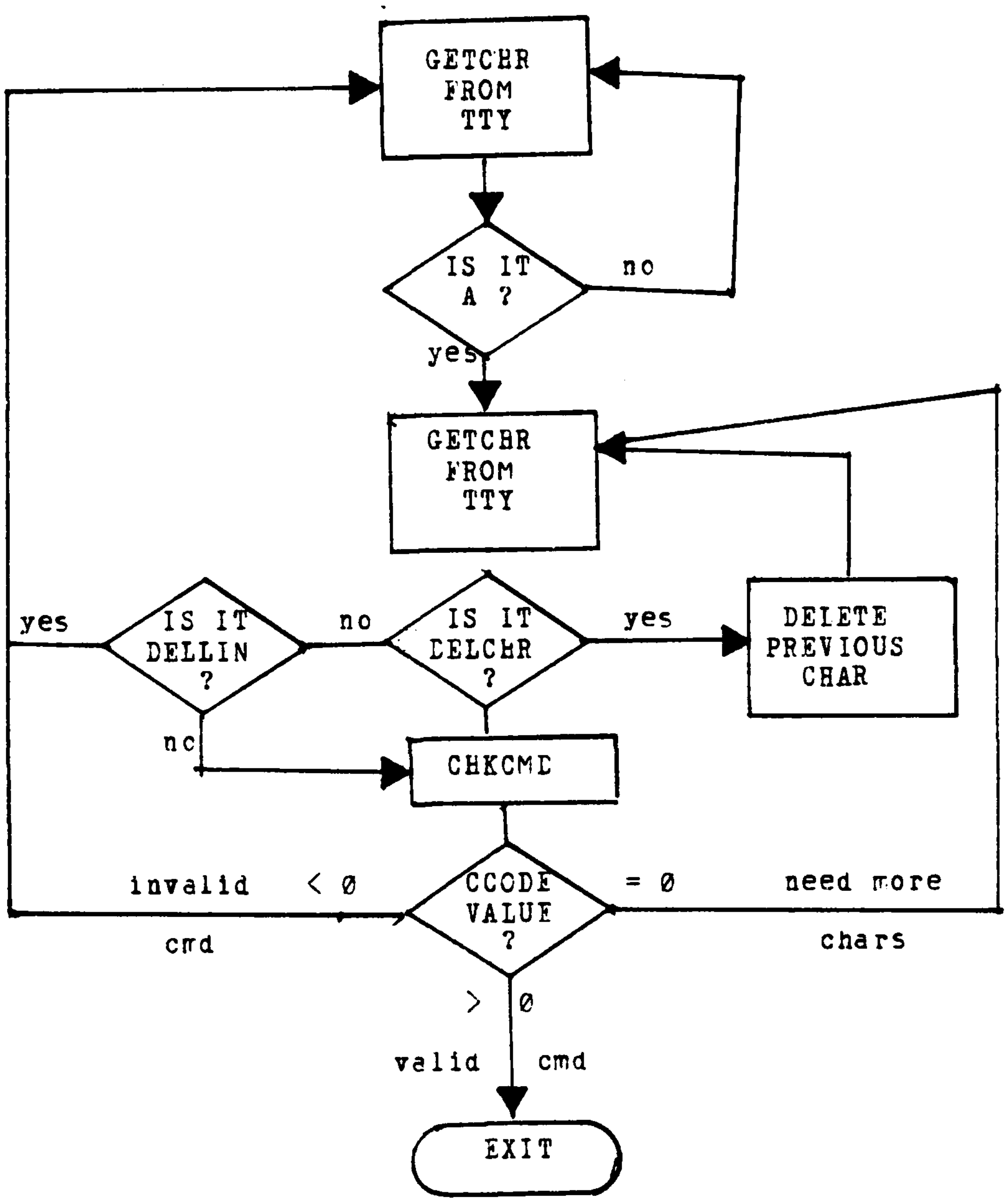


Figure 18. TTYIN Command Handling Logic

2. Data Structures

APPENDIX B contains a complete list of the parameters and global data structures used in the model. In this section, we will examine a few of the more important ones since familiarity with their contents and access methods is a prerequisite for an understanding of some of the program logic that we discuss in the next section. For better software management, all the global data structures are defined external to the executable code itself and are bound to the routines at compile time by use of the INCLUDE macro. This insures that each routine has an identical copy of the COMMON blocks it needs and eliminates the hard-to-find bugs that arise when this is not the case. This technique also makes it easier to change the programs, both during development and later during operational use.

For much the same reason, we made extensive use of the FORTRAN PARAMETER statement to define I/O units, array sizes, lengths of array entries, and other program constants. This improves program readability and, more importantly, simplifies the task of software maintenance. For example, to increase the number of events that the model can support, it is only necessary to change the value assigned to one of the symbolic constants (MAXEV) and recompile the program. In the discussions that follow, we shall frequently refer to these parameters both by symbolic name and by the constant value that they are assigned in the

AD-A115 757 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 17/2
AN INVESTIGATION OF THE APPLICATION OF VOICE INPUT/OUTPUT TECHN--ETC(U)
MAR 82 T R MALARKEY
UNCLASSIFIED NL

2 of 2
AD
A115 757

END
DATE
FILMED
7-82
DTIC

current version of the programs. For clarity, we will represent variable and parameter names in upper case, even though they are in lower case in the actual source files.

The most important data structure in the model is an array that holds the list of pending network events. This array (EVENTS) consists of four-byte entries, one for each future event. EVENTS is an ordered queue to which events are added in decreasing order of scheduled time. Thus, the last entry on the list is always the next event to be scheduled. An event entry contains the code for the event (EVCODE), the event subcode (SUBCOD), and the time (in seconds since the start of the program) when the event is to occur (EVTIME). The event codes are the ones listed in Figure 13; the event subcode will normally be the number of some network component such as an IMP or line. The following entry would trigger a message "IMP 2 is down" at 127 seconds after the start of the program:

```
EVCODE        SUBCOD        EVTIME
  11            2             127
```

Associated with the events list are a number of other global variables. NEV is the number of events on the list; TFNE is the time for the next event to occur, i.e. the EVTIME for the last entry on EVENTS; and TTNE is the time until the next event, computed by subtracting the current time from TFNE.

Some events, for example the "IMP DOWN" message, establish conditions that require intervention by the Network Controller. Such events are entered on the pending actions list (PAL) until the correct operator response is noted. When an event is pending, the program will repeat the associated ouput message every 60 seconds until an appropriate operator response is processed. PAL is an unordered array of four-byte entries structured identically to EVENTS. An entry remains on PAL until some appropriate operator action resolves it, at which time an entry is written in the model's log file (cncc.log) giving the EVCODE, SUBCOD, and the time in seconds that the entry was on the PAL. This file can then be analyzed to see if different operating modes have any effect on operator response times.

There are several data structures used by the voice output software. The two that have the most significance to the user, since their contents can be changed at run time, are two parallel arrays that associate network events with names of files where phoneme strings for voice output messages are stored. The first, VOMID, contains two-byte entries that denote the EVCCDE and SUBCOD for the current event. These are used as an index into the second, VOMSG, which contains expression identifiers used by the voice output software to locate the correct phoneme string for each message. These arrays, built from data in the disk file cncc.voc, can be modified by the user with any of the UNIX text editors. Figure 19 shows the contents of the file as it existed at NPS during the development of the model. The comments in the file describe the structure and contents of each entry.

There are five tables used to represent the topology and status of the COINS II network. ISTAT uses a single byte to denote the status of each IMP in the network using the standard status codes described in Section III. LSTAT performs a similar role for the subnetwork communication lines. MODLNS has the same structure as the topology table of the same name in the actual CNCC (see Reference 3 for a description). VHIMP and VHNAM contain one entry for each host in the network. A VHIMP entry contains the number of the IMP to which the host is connected; an entry in VHNAM contains the name of the host in ASCII. The network tables are initialized at the start of the program by reading them from disk. LSTAT and MODLNS are in the file coins.net; ISTAT in coins.imp; and VHIMP and VHNAM in coins.host.

There is only one important data structure used by ttyin, a table defined locally in the command handler. CMDS is a table that contains a four-byte entry for each operator command. The first byte is the number of characters in the command; the remainder contain the characters in the command mnemonic.

```
11 01 impdn##                   ! imp ## is down
12 01 linb##                    ! line ## is down
18 01 linp##                    ! line ## is down plus
18 02 linm##                    ! line ## is down minus
01 01 ierc##                    ! imp ## errors corrected
07 01 irel##                    ! imp ## reloaded
10 01 linup##                   ! line ## is up
10 02 linlp##                   ! line ## is looped
10 03 lindn##                   ! line ## down
14 01 bhdc##                    ! imp ## bd hst cksum
15 01 trap##                    ! imp ## trap
16 01 crash##                   ! imp ## crash

*       general format for an entry

*               cc nn name##

*       cc = event code associated with the message
*       nn = specific msg nr within this event
*       name = generic file name for this message
*       ## is component number for this instance
*       actual file name is the concat of name & Val(##)
*       program builds actual file name using current
*       value if subcod.
```

Figure 19. Contents of Voice Message Control File

3. Basic Event Handling Logic

CNCC is entirely event-driven in the sense that in the absence of events to execute, it would sleep forever. The model adopts a very generalized definition of "event"; any stimulus, whether internal or external, is considered an event under this definition. At the execution level, the event handlers perform their functions without regard to whether the stimulus came from inside the program or from the Network Controller. For purposes of discussion, however, we can categorize the events into three broad groupings. In the first group are those events that are scheduled automatically, either at initialization time or during operation; in the second are the responses to operator commands; and in the third is a hybrid set of events that combines features of the other two.

The first set consists essentially of those network events shown in figure 13 with the addition that certain summary reports, usually produced only on demand, are scheduled at the start of the program and later output automatically at the proper time. This illustrates the separation of the scheduling from execution that imparts a great deal of generality to the event handling routines. Typically this first set of events involves error conditions in some network component, such as an IMP, a line, or a host. At the start of the program, an event is scheduled for each network component. When this event is later executed, a new event is selected and scheduled. Both the specific event and the time for the it to occur are determined by the use of Monte Carlo techniques. The particular event is selected by comparing random numbers from a uniform distribution against a table that gives the cumulative probability distribution for the events of a component type. There are separate tables for IMPs and lines. The program makes the assumption that the time between component failures is distributed exponentially with some mean time between failures (MTBF) as the distribution parameter. MTBFs are defined separately for IMPs and lines and can be adjusted at runtime by changing the values in the file cncc.ini. The UNIX system only generates random numbers from a uniform(0,1) distribution. The program transforms a uniformly distributed random number (y) to an exponentially distributed number (t) by the formula

$$t = \frac{-\ln(y)}{\mathrm{MTBF}}$$

Analytic justification for this formula is presented by Gordon [Ref. 21: pp. 150-153].

Operator commands are executed on demand so there is normally no scheduling function to perform. Typically, the program will respond immediately to operator interrupts by performing some function and displaying the results on the Summary TTY. For certain types of events, those that require some time period to complete, a different strategy is required. The best way to describe this strategy is to consider an example.

Assume that you are the network controller and you want to send some local file to another COINS host, say SOLIS. You would type or speak the proper command and supporting parameters at the Summary TTY. At some time in the future you would expect to receive notification that the transfer was successful (or perhaps unsuccessful). The time to complete the transfer is a function of three variables: the size of the file, disk transfer time, and communications transfer time. The program first determines a figure for the size of the file (in 512-byte blocks) using a number from a uniform(10, 100) distribution. Then, using constant values for average disk transfer time (50 ms) and communication throughput (50K bits/sec), the program computes a value for average transfer time. This value is then used as the mean of a normal distribution whose standard deviation is approximated as 20% of the mean. A uniform random number is then generated and transformed into this normal distribution using an algorithm described by Gordon [Ref. 21: pp. 158-159].

This number is then used as the time to schedule the completion of the event. The completion event uses the same event code as the original operator command, in this case EVCODE 3; to distinguish between the two states, the SUBCOD field is made negative. When the event handler is called, it checks the SUBCOD and, if it is negative, knows that the event has been completed. It will then output the appropriate message. This same logic is used by a number of routines in the model, specifically the handlers for the DUMP, RELOAD, SEND, and BDCST commands as well as the event handler for the VIRTUAL-HOST-DOWN event.

4. Voice Output Software

There are two parts to the voice output software developed for the model. These parts correspond to the two steps involved in producing synthesized speech. The first step in the process is to convert the desired message text into phoneme strings that are recognizable by the synthesizer. The second is the recall and use of these strings as required by an application program. In a direct text-to-speech system these two steps would be merged; we prefer to think of them as distinct operations

The first stage of the process is supported by an interactive text-to-speech program called m11. This program accepts English text strings from the terminal, converts them to phoneme sequences, stores the sequences in a disk file, and sends them to the synthesizer to be spoken. The program then allows the user to iteratively refine the sequences using a full-screen text editor. The user can selectively tune the output strings by adding, deleting, or changing phonemes or modifying rate and inflection levels. Usually after a few iterations, the result will be fairly intelligible speech that can be reproduced by an application program.

The text-to-phoneme conversion routines were originally developed for the VAX-11/780 at The Naval Undersea Systems Center (NUSC) in Newport, R.I.; they were converted to UNIX F4p by the author. The conversion algorithm uses a set of rules that maps letters, syllables, or words into strings of phonemes. The particular implementation supports the VOTRAX ML-I; the design, however, is device-independent, since the text-to-phoneme conversion is achieved by a two-stage transformation process. Text is first converted to phoneme sequences from the International Phonetic Alphabet (IPA) which is an accepted standard of linguists and phonologists. A second routine performs the device-dependent conversion into the ML-I phoneme codes. It is likely, then, that the program would be fairly easy to adapt to other voice output devices.

Each output message is assigned a unique name of seven or fewer characters. The messages are stored in separate disk files to facilitate editing and recall, e.g. a message named "crash" is stored in a file named "crash.ml1." A library of FORTRAN-callable subroutines allows application programs to recall and output these messages on demand. The model uses the VOMSG table, described earlier, to select the name of the desired message and send it to the synthesizer.

E. PRELIMINARY RESULTS

Our intention in building the CNCC model was primarily to create a training and learning vehicle that might help us better understand the CNCC application and the possible utility of voice technology. We were not looking to develop a formal method for quantifying this utility. The only variable that the program attempts to measure is the time it takes for the Network Controller to respond to certain events. As we noted earlier, a formal evaluation experiment was not a part of our overall strategy. Despite this, we decided to use the model to obtain some idea of the way that different input and output modes affect operator response time and productivity.

The methodology that we adopted was to run the model for approximately two-hour periods under three different environments. We first ran in a mode where input commands were typed manually and alert messages were displayed on the Logger TTY, accompanied by bells that functioned as audible alarms. Next, we used the same output mode but spoke the commands to an automatic speech recognition device rather than typing them manually. Finally, we added voice output to this configuration by sending the alert messages to the ML-I speech synthesizer to be spoken aloud. While the model was running, the author, playing the role of the Network Controller, worked on a parallel task a short distance from the Summary and Logger TTYs. (This parallel task was, incidentally, the production of this thesis.) When the model issued an alert that called for operator action, he would issue the appropriate command by either typing or speaking it, depending on the particular operating environment.

The results, in terms of operator response time, are summarized in Figure 20. As the results show, there was a dramatic reduction in average response time when we replaced typed commands with spoken ones and an additional, but less dramatic, reduction when we added voice output. The amount of variability in response time, as measured by the standard deviation, follows a similar pattern.

Perhaps more meaningful would be some indication, however subjective, of the effects of the different modes on parallel-task productivity. In the first scenario, very little was accomplished on the parallel task since the operator seemed to be constantly moving from his text-editing work station to either the Summary or the Logger TTY. The actual duration of a typical interruption was generally fairly brief. The effect, however, was usually much more significant since the time would be long enough to cause the operator to lose his train of thought by the time he returned to the task. In the second operating mode, response times were substantially better; but the improvement did not carry over as much to the parallel task. There was still a need to walk to the Logger TTY and read the alarm message before issuing the proper recovery

OPERATING MODE	NUMBER OF RESPONSES	RESPONSE TIME MEAN	STD DEV	RANGE
1	35	35.6	26.7	8-138
2	34	12.8	16.5	4- 72
3	36	8.6	11.7	3- 68

OPERATING MODES

1. Manual input, printed output.
2. Speech input, printed output.
3. Speech input, speech output.

Figure 20. Operator Response Times (in seconds)

command. In the third situation, the interruptions seemed to have much less of an effect. There was no need for the operator to move away from the terminal where he was doing his text-editing. He could hear the error messages and could quickly formulate a response, speak the appropriate corrective command, and return to work.

These results must certainly be viewed with a cautious eye. First of all, since no formal experiment was conducted, even the quantitative results lack the solidity of those that result from a controlled and replicated experiment. They permit no inference as to the statistical significance of any of the variable factors. Secondly, the subjective comments on parallel-task productivity have to be considered unsubstantiated observations from a possibly biased source, i.e. the author. Lastly, it is not certain that faster response time and better parallel productivity are, by themselves, all that important. The Network Controller needs to respond quickly to network events, but is there really a substantive difference between 35 seconds and 8 seconds? Is overall COINS operation significantly improved by such a response-time reduction? Improved parallel productivity implies that overall operator effectiveness will be increased, but is the improvement sufficient to justify the cost of the voice technology? These questions cannot be answered by merely running a model and examining the results. They require a careful management appraisal before a decision can be reached that voice technology in the CNCC is or is not cost-effective. Presumably, the principal value of the CNCC model is that it can play a useful role in that assessment.

VI. APPROACHES TO IMPLEMENTATION

A. GENERAL

Having demonstrated both the applicability and the feasibility of voice technology in the CNCC, we turn now to a discussion of how this technology might be installed in the existing Network Control Center. In this section we will consider both the approximate cost and the steps involved in implementation. Although we will examine the merits and shortcomings of existing systems, we stop short of recommending the hardware of any specific manufacturer. The commercial market for voice technology is highly volatile and the system that seems the best choice now might not be best in six months. If it is decided to go ahead with voice input or output for the CNCC, a comparison that explores the tradeoffs between cost and capability will be necessary before buying specific hardware. Presumably, the discussions in this section can serve as a starting point for this effort. The ultimate decision on the utility of voice technology will,of course, be based on its cost-effectiveness to the COINS project as a whole.

Voice input and output are related but, in many ways, entirely different technologies that raise different implementation questions. We will therefore address them separately in this section. We will first look at the implementation of Automatic Speech Recognition, where we have accumulated a good deal of experience and where implementation is straight forward. We then move on to the murkier area of speech output where we have much less direct experience and where implementation is more involved, with a number of different strategies available.

B. VOICE INPUT

1. CNCC Voice Input Requirements

Installation of ASR devices is so simple as to make the term "implementation" sound a little pretentious. All that is involved is connecting the recognizer to the host (in this case the CNCC) via an RS-232 interface, developing and training the vocabulary, and proceeding to use the system. The vocabulary that we used to test the CNCC model can serve as the basis for building the one to be used in actual operation. The main problem is in selecting the particular system to use. This process, not surprisingly, involves the classical tradeoff between cost and capability.

Our experience to date suggests that a speech recognizer used in the CNCC should have the following minimal set of features:

-- support a vocabulary of at least 100 utterances;

-- be able to operate in a moderately noisy environment;

-- support a wireless transmission capability; and

-- have recognition accuracy of over 95%.

The test vocabulary that we used with the CNCC model contains about 86 utterances. At least thirteen of these, however, would not be used in the CNCC in actual operation. Examples include: "Stop the Model", "Change directory to COINS" and several others that were included more for convenience than operational necessity. We did not implement the entire set of CNCC commands in the model and there will be a number of utterances that we will undoubtedly want to add. A 100-utterance vocabulary should be viewed as a lower-bound; 150 or 200 might be really more practical to allow for vocabulary expansion.

Computer centers always generate some amount of background noise. The banging of printers, the whirr of air conditioners, and the opening and closing of doors and drawers are heard routinely in any ADP operations area. It is necessary, therefore, that the speech recognition system use some technique to filter out the unwanted background noise. Usually this is accomplished by use of a specially-designed noise-cancelling microphone.

As we pointed out earlier, the Network Controller is frequently away from the Summary TTY console. It would be inconvenient for him to return to the console to enter voice input commands. It would be equally inconvenient, as well as unsafe, to use a headset with wires trailing back to the voice input unit. There is a definite requirement, therefore, for some type of radio transmitter to pass signals from the microphone to the recognizer. This will permit convenient and safe operation at some distance from the Summary TTY.

A reasonable level of recognition accuracy is essential for successful operational use. To be beneficial, speech recognition should reduce the number of input errors that would occur if commands where entered manually. A number of manufacturers of moderately-priced recognizers advertise recognition rates in the 95% range. These figures can be misleading, however, because recognition accuracy is very much a function of vocabulary design and experience of the users. Marketing brochures, therefore, should be looked at rather carefully, the best guarantee of recognition accuracy being a test with the desired vocabulary. Based on experience with the CNCC model, there should be little problem in attaining an acceptable accuracy rate with several of the recognizers available today. It is when one begins to look for error rates of 1% or less that the slopes of the cost curves become very steep.

Several authors have commented on an interesting behavioral phenomenon exhibited by people being introduced to voice input for the first time. Users are much more critical of voice input errors than of errors in typing. Poock [Ref. 8: pp. 21-22] noted that when he demonstrated various software products at the Naval Postgraduate School, users accepted his frequent typing errors as a matter of course. However, if he made one error while using voice input the view often expressed was, "voice input is nice, but it's obviously not perfect and has a long way to go." While we obviously want an ASR system to be as accurate as possible, we should guard against adopting a "double standard" between speech and typing.

2. Possible Voice Recognition Systems

In examining alternative voice recognition systems available in the commercial market, we focused on what could be considered the middle range of voice recognition products(roughly $8,000 to $20,000 for a suitably equipped system) and tended to ignore systems at either end of the price spectrum. Scott Instruments, for example, markets a complete voice recognition system (the Vet/2) that runs with either the Apple II or the TRS-80 for about $900 [Ref. 22: p. 106]. The evidence so far suggests that the Scott recognizer and the oher low-priced systems do not meet either the vocabulary size requirement or the acceptable accuracy level. At the high end of the spectrum, Verbex and Nippon Electric both manufacture systems with excellent accuracy ratings that are designed to handle limited connected speech. These systems retail for over $65,000, however, a price that seems much too expensive for the CNCC application.

Of the many commercially available speech recognition systems in the desired price range, there are at least two that promise good support for the CNCC application. These are the Threshold T-600, manufactured by Threshold Technology of Delran N.J. and the Interstate VRM, produced by Interstate Electronics Corporation of Anaheim, Ca. The reader may gain some insight into the rapidity of change in the voice technology market by the following item. When this section was first being drafted, there were three systems that seemed to have potential for the CNCC application. When the author phoned the third (unnamed) company for information, he was informed that the company, previously regarded as one of the leaders in the field, was no longer making voice recognition equipment.

The Threshold-600 was the voice recognizer used with the CNCC model in the NPS laboratory. It was quite easy to use and performed very reliably. The T-600 system is made up of a Shure SM-10 noise-cancelling microphone, an Ann Arbor large character display and operator console, a tape cartridge unit, an analog speech preprocessor, an LSI-11 microcomputer, and a serial RS-232C compatible input/output interface. An FM wireless transmitter is available for an additional charge of about $5000. The recognizer will accept as many as 256 discrete utterances of up to two seconds in duration. A pause of at least 100 ms. is required between utterances. The processing time to recognize an utterance depends on the size of the vocabulary but will usually be no more than 250 ms. [Ref. 23]

In terms of performance, the T-600 is one of the best in the field. The founder and President of the company, Thomas B. Martin, is one of the pioneers in developing practical speech recognition systems. The company has a wide and varied customer base and its products have been used in almost every conceivable application. In addition, the T-600 has been used in several intelligence and Command and Control contexts and the experience gained from these operations should be transferable to an application such as the CNCC. All in all the T-600 represents mature technology from a stable and experienced company. The principal disadvantage of the T-600 is its fairly high cost in comparison to some of the other recognizers on the market. A fully equipped T-600, with the wireless FM transmitter, will cost in the neighborhood of $15,000.

The Interstate Voice Response Module (VRM), by contrast, can be purchased for as little as $5,000, depending on the options desired. The company does not explicitly advertise an FM transmitter, but assuming that one were available at a price comparable to that of the Threshold wireless system (about $5,000), the total cost of an Interstate system would be roughly $10,000, about 50% less than for the T-600. The Interstate Voice terminal

(Voterm), combines a keyboard, display, 48K bytes of Random Access Memory (RAM), a Z80 microprocessor, and a 100M-byte floppy-disk drive. The Voterm will recognize up to 100 speaker-dependent discrete utterances. An additional feature worth noting is a board (available for about $1000) that provides a voice response capability. The board, which uses VOTRAX's SC-01 chip, supports 500 words stored in memory and the ability to program about 1000 additional words.

While it possesses some attractive and potentially useful features, the VRM does have at least two possible drawbacks. First, the 100 word vocabulary may leave insufficient room for expansion and might not provide enough phrases for the CNCC operation. Second, there is little experience within the C3 or intelligence communities with the VRM. The author did visit an experimental effort at NASA's Ames Research Center that used the Interstate Voterm. However, this particular effort used a very small vocabulary (only four words) and thus it is hard to draw a comparison with the CNCC where a much larger vocabulary is required. The NPS Human Factors laboratory has plans to purchase a Voterm this spring, and their experience may help us to evaluate the utility of the Interstate VRM in the CNCC.

3. Potential Areas of Concern

Most of our discussion in this thesis has emphasized the positive aspects of Automatic Speech Recognition. There is no question that we believe it to be a technique that can assist the Network Controller in performing his duties more effectively and conveniently. Nonetheless, as with any technology that introduces fundamental changes in operating procedures, it is not without some risk. There seem to be two issues that should be carefully considered before proceeding with installation of a speech recognition system. We will discuss each of them briefly here, but they clearly warrant further consideration by the CCINS PMC and CNCC management.

The first question concerns the use of voice recognition by more than one speaker on each shift. As we have discussed elsewhere in this thesis, voice recognition systems are speaker-dependent and must be trained by each user of the system. In other words, each Network Controller would train the system with his voice patterns and then store these patterns on a small cassette. At the start of a shift, the Controller would read the cassette into the memory of the recognition system (which takes less than a minute) and would then be ready to issue voice commands. This scenario presumes, however, that only one operator per shift will use the voice equipment. If this assumption is not valid, some alternate procedure will have to be adopted. One possibility, if the vocabulary is small enough, is to train more than one operator on a single tape. This approach has been tried informally at NPS and appears promising. Depending on the size of the vocabulary and the ingenuity used to design it, we might be able to store as many as three sets of voice patterns on a single tape cassette. The problem is not insurmountable; it simply has to be addressed before voice recognition equipment is selected and installed.

The second concern stems from the fact that our study of the CNCC was conducted second-hand at a distance of 3000 miles. We have attempted to digest and understand all the operating procedures for the CNCC. We are not naive enough to believe, however, that manuals of operating procedures always accurately mirror the operation that they purport to describe. For many of the situations that arise in the COINS II network, the CNCC operators manual prescribes alternative courses of action. For example, corrective procedures can often be initiated from either the Summary TTY using the CNCC command language or the Master IMP using a different set of commands. Our analysis and subsequent model have made the, perhaps simplistic, assumption of a single data entry point -- the Summary TTY. To the extent that this assumption is reasonable, a single voice recognition device will be workable. Again, it may be possible that the convenience offered by voice recognition might cause the procedures themselves to be modified so that the assumption of a single command entry position would be more realistic. We raise these question merely to generate thought and discussion so that they and any other questions that may arise because of the installation of a voice recognition system can be resolved as early as possible.

C. VOICE OUTPUT

1. General

With voice recognition, there were really no implementation issues other selection of the best equipment at the lowest cost and the need for careful planning. For this reason, we focused on requirements and the different capabilities of two possible ASR systems. When we turn to the output side, we encounter a different problem. With output, the requirements are simple and can be satisfied with a whole range of relatively inexpensive products. The problem that arises concerns the way that a voice output device is connected to the CNCC. As we noted earlier, voice output is software-intensive. The fundamental design question, then, becomes one of where to locate this software to minimize the cost and overall impact. We are interested less in particular commercial products than in the broader issue of implementation strategy.

There are only two fundamental requirements of a voice output system in the CNCC. First, it should be capable of producing speech output from prestored messages that is intelligible to someone who is familiar with the output vocabulary. Second, implementation should require no major software or hardware modifications to the existing CNCC, and preferably none at all. There is no requirement for a real-time text to speech capability. Nor is there even a need for "human sounding" speech. A voice output alarm would be in addition to rather than a replacement for the Logger TTY. Even if the voice output messages were not perfectly intelligible, the system would still be at least as good as the existing one. we do, of course, want to get the best system possible within some cost range; it can, however, be effective even if voice quality is not the highest. Voice response systems are available on the market for prices as low as $400 up to more than $10,000, with most under $5,000. Based on our understanding of the CNCC operation and our experience with the CNCC model, we see no reason to go above the middle range.

We will consider three alternative implementation strategies. They differ mainly in the location of the text-speech software and in overall flexibility.

a. The voice output terminal functions as an unintelligent peripheral device connected directly to the CNCC computer. It performs no computing functions other than the conversion of the phoneme strings to output speech.

b. The voice output device is a stand alone voice response computer that would accept ASCII-coded character strings from the CNCC and transform them to voice output messages.

c. The synthesizer is a slave peripheral, as in the first strategy, but its host computer is not the CNCC but a general-purpose microcomputer that would handle the text-speech transformation and other computing functions.

In this thesis we are more concerned with overall system architecture than with evaluating specific commercial products. In the discussions that follow, however, we have found it necessary to establish some benchmarks so that we can ascribe approximate costs to the different alternatives. For the first and third alternatives, our benchmark system is the VOTRAX Model SVA synthesizer manufactured by Federal Screw Works of Troy Michigan. The SVA is an improved version of the company's popular "Type-'N-Talk" text-to-speech converter. Both use the VOTRAX SC-01 synthesizer chip, but the SVA has four times as much memory and better control over speech inflection; it costs about $2,000. The VOTRAX ML-I, which we used in the C3 lab, does not seem to provide enough extra capability to warrant its $10,000 price tag, at least as far as the CNCC application is concerned. For the second option, our benchmark system is the SLC-II Intelligent Communications Controller manufactured by Digital Pathways Inc. of Palo Alto, California. The SLC-II is a stand alone computer with voice output capability built-in; it sells for about $3,200.

2. Slave Voice Output Peripheral

This design approach, which is the one used for the CNCC model at NPS, views the synthesizer as a black-box whose sole function is to convert phoneme strings output by the host computer into intelligible speech. Figure 21 shows the synthesizer connected to the Logger TTY in what is referred to as a Shared Channel Configuration (SSC). Alternatively, if there were a sufficient number of serial interfaces, the synthesizer and the Logger TTY could be on separate RS-232C data channels. The burden of somehow generating the phoneme strings is borne by the host computer, in this case the CNCC. We should note that the SVA does have fairly good text-to-speech capability that, in theory, would mean that it could translate the existing Logger TTY messages directly to speech. While we recognize this possibility, it does not seem that this alternative is very practical. For one thing, text-to-speech algorithms are not perfect and, based on our experience, the results will probably not be satisfactory. English is not regarded as a very phonetic language and text-to-speech synthesis by rule might cause the word "IMP" to be pronounced "Eye-m-p" or the word LINE to come out as "L-ee-n". To get the necessary control over pronunciation, the phoneme strings should probably be prepared in advance. Thus, although we

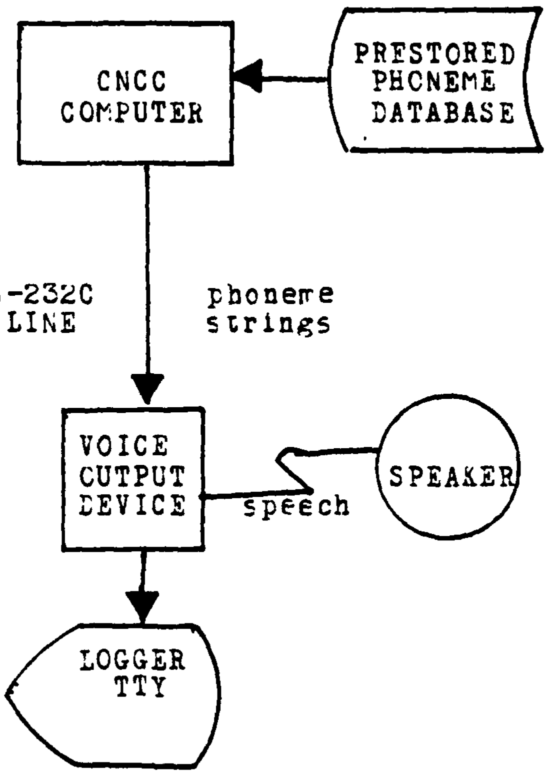


Figure 21. Voice Synthesizer as CNCC Terminal

acknowledge realtime text-to-speech conversion as a possibility, we think it more realistic to consider the worst case, i.e. where the CNCC must output the phoneme strings.

There is clearly a serious drawback to this approach in that it requires software changes in the CNCC. It is worth considering, however, how we might implement it in such a way as to minimize the amount of change required. First of all, the original text-to-phoneme conversion should not take place on the CNCC. A modified version of the conversion program developed at NPS could be run on a COINS TAS to produce the phoneme strings. These would be moved to the CNCC and stored on disk, requiring about 15,000 bytes of on-line storage. The CNCC program would be modified to recognize the situations when a voice output message was required, read the appropriate phoneme string from disk, and output it to the synthesizer. As a gross estimate, this would require at least 500-1000 words of main memory for code and buffers.

It is difficult to estimate the cost of the CNCC software changes, or whether in fact they can even be accomplished. Based on experience, however, it is hard to believe that the cost would be much less than $10,000, assuming the work was done under contract. This means that the cost of this option would be at least $12,000 and could very possibly be higher. About the only real advantage to the approach is the fact that we can use the text-to-speech editor that we developed at NPS with only slight modification. In that case, however, the host computer was considerably more powerful and flexible than the Honeywell-316. All in all, this alternative is potentially very costly and there is some risk that it can't even be done at all. It is of interest more for academic than practical reasons.

3. Stand Alone Voice Response Computer

This design, depicted in Figure 22, eliminates the need for any software modifications to the CNCC computer. Messages would be sent to the Logger TTY as they are currently but would actually be read by a microcomputer such as the Digital Pathways SLC-II. This computer would pass on the messages to the Logger TTY but would first scan them to detect those that should have an associated audio response message. For these, the microprocessor would retrieve and speak the appropriate output message.

The SLC-II is a microprocessor-based communications controller that can support up to 5 serial I/o channels. The built-in speech synthesizer can be used to announce events over the unit's front panel speaker or via the telephone. It is usually taught the details of a specific application by means of simple commands to its own operating system called SAMSYN. The profile of the application is stored in battery-supported memory so that, after the

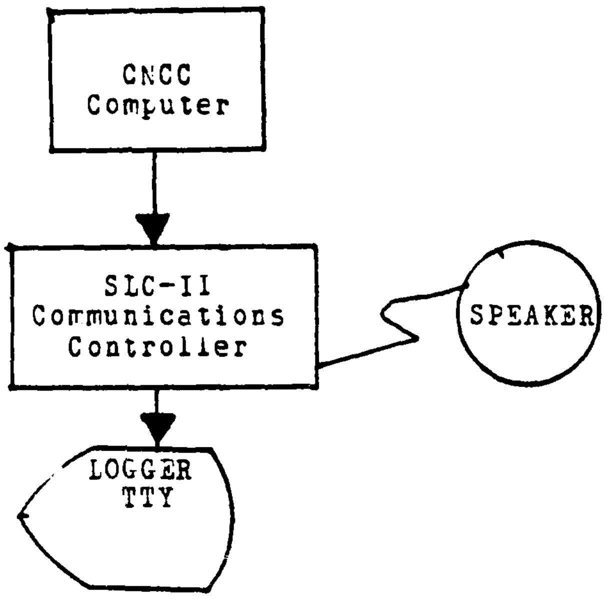


Figure 22. Stand Alone Voice Response Computer

initial training session, the unit can be considered to be a 'black box' tailored for a specific application. Along with the basic unit and the operating system, the SLC-II is equipped with a version of microsoft BASIC to extend its software capability. The SLC-II uses a proprietary bank-switching technique that allows it to access up to 80K bytes of RAM [Ref. 24: p. I-1].

The SLC-II would be interfaced to the CNCC computer and the Logger TTY on serial RS-232 or 20 mA channels. SAMSYN provides the capability to define KEYS that are ASCII strings that the SLC-II looks for on its serial line from the host. Associated with each KEY is an ACTION. If a KEY is detected in the midst of the serial data stream from the host, it immediately triggers an ACTION; KEY n triggers ACTION n. ACTIONs cause particular programs to be executed, i.e. they may dial phone numbers, speak sentences, etc. Typically, in the case of the CNCC, the action would involve speaking a sentence such as "IMP 5 is down." Sentences are composed from a pre-stored vocabulary.

The SLC-II has several attractive features. Fully equipped, with 80K bytes of memory and all software, it costs $3200. Moreover, it requires no user software as such, just the definition of KEYs, ACTIONs, and SENTENCEs which can be done by someone who is not an experienced programmer. It is compact, with a built in speaker, and its user manual is clear and readable.

There are two potential drawbacks. First, the voice vocabulary that is provided is not defined in the manual, so there is no way of knowing whether or not it is adequate. The vocabulary requirements of the CNCC are, to be sure, small, but a number of uncommon words are used. It is possible that, if necessary, an extended vocabulary could be purchased at some extra cost. Potentially a bigger problem is the limited number of KEYs and ACTION's, currently set at 16 each. For the model at NPS we used only 12 basic sentences. Each had a number of variants, however, for each of the applicable network components. There were six variants of the "IMP down" message, for example, one for each IMP in the CCINS subnetwork. There is some suggestion in the user's manual that these limitations might be dealt with by invoking user-callable subroutines which use special ASCII strings called buffers. Buffers seem to provide a way to formulate an speak sentences dynamically. The comparatively low cost and the ease of implementation suggest that these issues should be investigated further.

4. Microcomputer-Controlled Synthesis

This architecture, shown in Figure 23, is essentially an attempt to combine the advantages of the first two designs while minimizing their drawbacks. It shares with the first the advantage of a convenient and

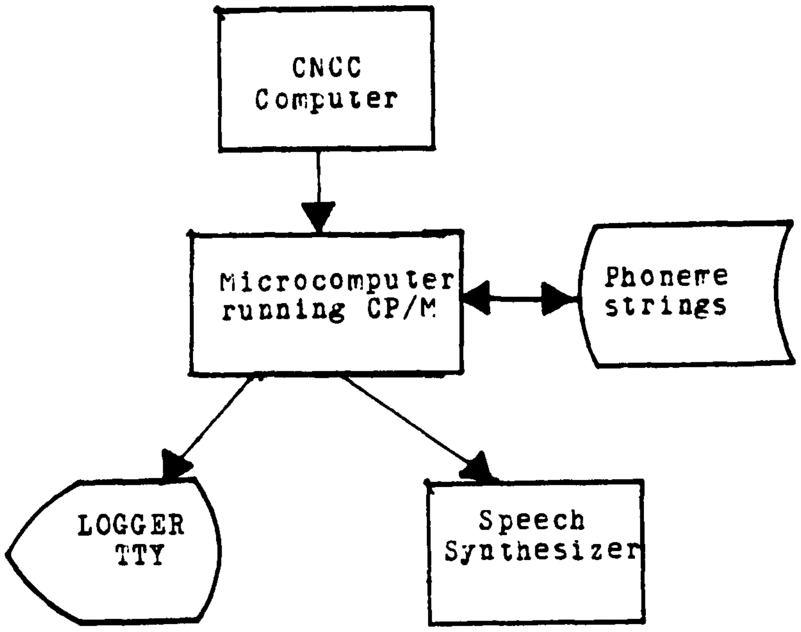


Figure 23. Synthezizer Controlled by Microcomputer

powerful text-to-speech editor and, with the second, the advantage that no changes to CNCC software are required. Additionally, it is more flexible than the second approach because it makes available the full capabilities of a standard general-purpose operating system.

The system would work in the following way. The user would employ a special full-screen editor to generate the phoneme strings to drive one of the VOTRAX speech synthesizers such as the SVA described above. These strings would be stored in standard CP/M text files. An application program running under the CP/M disk operating system would capture messages sent by the CNCC and pass them to the Logger TTY. It would also scan each message looking for alarm conditions. Once the program recognized an alarm message, it would retrieve the corresponding phoneme string and send it to the SVA.

A speech editor that could be used to support this strategy was developed by Psycho-Linguistic Research Associates of Menlo Park, California and is marketed under under the trade name SpeechWizard. SpeechWizard is a full-screen editor for developing phoneme codes for the entire family of VOTRAX synthesizers. It is not a text-to-speech program but rather a speech editor that generates codes for stored vocabulary to be used later in application programs. In that sense it is similar to the mil editor developed on UNIX for the CNCC model. It can be run on any microcomputer that supports the CP/M operating system, which includes virtually every major personal computer. Using a special SOUNDSPELLING system, the user can generate voice messages without learning about acoustic and articulatory phonetics and without dealing with hexadecimal, octal, or ASCII codes. According to the developer, Dr. Carol A. Simpson, SpeechWizard has been tested with linguistically naive users who have used it to create intelligible speech [Ref. 25].

SpeechWizard is being used at NASA's Ames Research Center to support a research effort designed to assist helicopter pilots flying low-altitude missions. The system, installed on a SCL-20 8080-based microcomputer, seems easy to use and the output speech, produced by a VOTRAX ML-I, is clear and intelligible. The price for SpeechWizard is $325; of course to use it requires a microcomputer that runs the CP/M disk operating system. The cost of such systems will vary but a fairly well-equipped system can be purchased for about $4,000. Assuming that we used the VOTRAX SVA synthesizer,the total cost would be in the $6,000-$7,000 range.

Implementation of this alternative would also require the development of a program to scan the CNCC messages and select the appropriate voice response based on the contents. This program can be written in a higher-level language and should be relatively straightforward since it is essentially a straight character string match and table lookup. Someone familiar with the CNCC and CP/M should be able to develop such a program in a week or less. Although it costs a little more than the second alternative, this strategy provides a good deal more flexibility and possibility for enhancement. The number of possible output messages has no practical limit and the quality of the output speech is good. Since SpeechWizard was developed by a professional linguist, it embodies a considerable amount of linguistic knowledge. The availability of a programmable microcomputer also opens up other possibilities. The microcomputer could, for example, reformat some of the CNCC Summary reports to improve their readability. It also could perform some local command editing and provide voice feedback to the Network Controller. While we are not really looking for a way to improve other aspects of CNCC operation, if the possibilities for such improvements are presented, it would seem shortsighted to ignore them.

D. COMBINED VOICE INPUT AND OUTPUT

Although voice input and output represent two independent technologies, it is natural to want to combine them in a single system. In several places in this thesis, we have alluded to a closed-loop system combining speech recognition and voice response. Implementation of such a system would not be easy, particularly in view of the justifiable reluctance to make any modifications to the CNCC software. Nevertheless, a step in that direction might be possible if we could install a system connected to the CNCC that supported both speech input and output and also allowed for the development of user software to employ them intelligently.

The Interstate VOTERM, which we discussed above in connection with voice input, provides at least the possibility to construct such a system which might be configured as shown in Figure 24. The VOTERM is Z80-based and provides 48K bytes of RAM, an 100M-byte floppy-disk drive, a keyboard and display screen, the discrete-utterance Voice Recognition Module (VRM), and voice response from a 500-word vocabulary through a VOTRAX SC-01 chip, the same one used in the SVA synthesizer. Base cost for the system is about $11,000, $6,000 for the VOTERM itself and about $5,000 for a wireless transmitter; an expanded model might cost as much as $15,000. This is in contrast to a cost of over $20,000 if we used a T-600 speech recognition system with a microcomputer-based voice output device. As we noted above, we don't have much direct experience with the Interstate product and it may come up short on some of the CNCC requirements. Its 100-utterance recognition vocabulary, for example, may not be adequate for CNCC use. The ability to do write user programs in high-level languages may provide a means to overcome these limitations. In light of its relatively low cost and in view of the

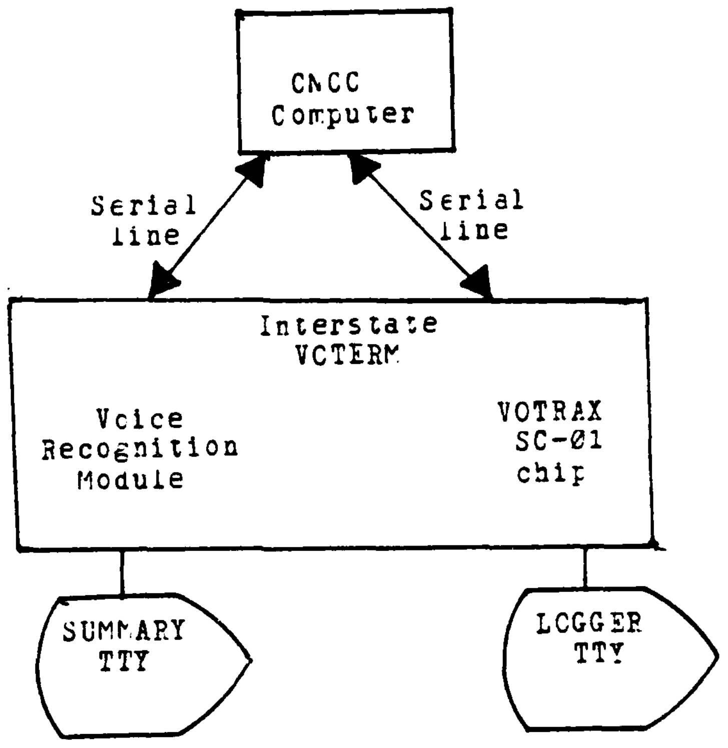


Figure 24. Combined Voice Input/Output System

functional advantages of combining input and output a single unit, this strategy is certainly worth pursuing further.

VII. SUMMARY AND CONCLUSIONS

We have taken several different paths in the preceding discussions; it seems in order, therefore, to try to bring these paths together and summarize our findings. We first developed a profile of common characteristics of applications that have successfully used voice input. The CNCC application scored fairly high when matched against this profile. In a less formal way, we looked at typical applications of voice output and again concluded that the CNCC was a reasonable candidate for application of this technology.

Having established the potential usefulness of voice technology in the CNCC, we proceeded to develop a computer model that would help us evaluate the feasibility of VIO and, at the same time, give a preliminary indication as to the contribution that it might make to the overall effectiveness of the Network Controller function. Our conclusions in this area were rather tentative. We showed, for example, that use of voice technology resulted in marked improvement in operator response time for one informal test that compared the different operating environments. This same test seemed to suggest that the use of voice technology, particularly voice output, would increase the productivity of the Network Controller in performing parallel tasks. Again, caution must be used in interpreting these results since the tests were informal, unstructured, and unreplicated. The conclusion with regard to parallel-task productivity, in particular, must be viewed as the subjective impression of a single test subject with no quantitative measure to support it.

The applicability and feasibility established, we turned to the consideration of how best to integrate the technology into the existing CNCC. Installation of Speech Recognition equipment was shown to be straight forward, involving no changes to the current hardware or software configuration. At least two manufacturers offer systems that would probably meet the needs of the Network Controller function; selection of the one to use involves the classical tradeoff between cost and capability. The installation of a voice input device would necessitate modification to some of the current CNCC operating procedures. As with any change to an existing operating environment, careful planning is essential in order to maximize the benefits and minimize the potential disruption.

The probability of operational disruption is much lower for voice output. The addition of spoken messages, even if the speech quality is not too high, is unlikely to hamper operations. The most serious problem with voice output, at least at this time, is that it requires considerably more software than does speech recognition. To implement an audio response capability without software modifications to the CNCC will require that the voice output device be driven by an external computer, either a self-contained Voice Response Computer or a general-purpose microcomputer. Whichever approach is chosen, a certain amount of software development and maintenance will be required. In this context, "software" includes anything from creation and editing of voice messages to development of simple programs in a higher-level language like BASIC. The key point is that, while we may be able to minimize the amount of software work, we cannot eliminate it all.

In the section on implementation, we included some discussion of the comparative costs of the different alternatives. This is a difficult area to summarize since, as might be expected, the cost picture is a comparatively fluid one. It is possible, nonetheless, to at least establish some cost boundaries as guidelines for decision-making. At the low end, it is possible to implement a combined voice input/output system for as little as $11,000; other potential configurations could cost twice that much, and there are a number of alternatives in between. Our investigation has shown that the technology can improve operations in the CNCC. The question of whether or not the improvement is worth the cost is, quite properly, a management rather than a technical one. We hope that some of the groundwork done in this thesis might make that question easier to answer.

APPENDIX A

OPERATING INSTRUCTIONS FOR CNCC MODEL

These instructions assume that the person who wants to run the model has a working familiarity with both the UNIX operating system and the CNCC operating procedures as outlined in The CNCC Operators Manual [Reference 3]. Given these assumptions, running the model is fairly straightforward. The process involves three steps: first, ensuring that the support files contain the correct information; second, running the model; and third, analyzing the results.

There are a number of files that the system uses to initialize internal parameters and tables. The casual user will only ever have to make changes to one of them, the file cncc.ini. This file contains a number of parameters and switches that the program uses while operating. The file is a standard UNIX text file that can be modified with any of the text editors, such as the full-screen RAND editor. The file also contains comments that assist the user in making changes. The contents of the file as used at NPS are as follows:

```
48.0,86400.0,86400.0,-1,/dev/ttyh
rtmul = 48.0    ; nr of times real time for this run
imtbf = 86400.0 ; mtbf for an imp in secs
```

```
lmtbf = 86400.0 ; mtbf for a line in secs
voflg = -1      ; voice output enabled
logdev = /dev/ttyh ; device for voice output
```

The parameter fields contain numbers or character strings and are delimited by commas; the format is fairly rigid and all fields except the last must contain the required number of character positions. The first parameter (RTMUL) is the multiplication factor that the model uses to speed up real time. In the example, each half hour of program time would correspond to an entire day of real time. The parameters IMTBF and LMTBF represent the MTBF for IMPs and lines respectively. In this case, MTBF for both is one day (86,400 seconds). The program will apply the real time multiplication factor to these values converting them, in this case, to 30 minutes each. The parameter VOFLG determines whether or not voice output messages will be used. In the example, the value of -1 turns on voice output; to disable it, the user would change the value to 0. The last field contains the UNIX path name for the device that is to serve as the Logger TTY, in this case /dev/ttyh. This provides flexibility in configuring the system. Note that as the system is currently designed, the voice output device and the Logger TTY must be the same terminal.

Once the user has made any required changes to cncc.ini, he or she is ready to run the model. To do this, he should LOGIN on both the Summary and Logger TTYs with the same USERID on both devices. Next, change the working directory on both terminals to the directory where the software for the model resides. (At NPS, this is currently /usr/malarkey/coins.) Next, on the Logger tty, enter the command 'back'; this will start up a background process that will periodically type out the current date and time. Then, on the Summary TTY, enter the command 'cncc' and the model will start to execute. The program will first type out the current date and time on the summary TTY and then will be ready to accept command input. The following commands are currently supported: REPORT, BROADCAST FILE, PRINT DIRECTORY, PRINT TOTAL, SEND LOCAL-FILE, DUMP, RENAME FILE, and DELETE FILE. See the CNCC Operators Manual for details.

In addition, two special commands have been included for operator convenience. It may be useful for the operator to change the status of one of the lines in the network. To do this, a special command has been implemented. The user types '?l'; the system responds with 'INE STATUS IS'; the user then enters either a 'd', a 'u' or an 'l' to denote that the line is either down, up, or looped; the system then responds 'LINE: ' and the user enters the line number followed by an escape character. Here are two examples:

```
?lINE STATUS IS uP  LINE: 2$
?lINE STATUS IS lOOPED  LINE: 14$
```

Typically this command is used to restore a line that has been marked down by some network event. The other special command provides a way to gracefully stop the model. To halt, type '?q'; the system will respond 'UIT CONFIRM (Y OR N)' and the user responds appropriately.

After running the model, the user may wish to examine the response times for the run. These results are in a file named cncc.log. Each line in the file corresponds to one event that required operator action and contains the event code, the subcode, and the number of seconds that elapsed between the time that the failure occurred and the time that an applicable operator response was noted.

APPENDIX B

PARAMETERS AND GLOBAL DATA STRUCTURES

1. PARAMETER DEFINITIONS

```
parameter evlsz = 200           ! size of event lst
parameter maxev =  50           ! max nr of events
parameter leve  =   4           ! length of event entry
parameter palsz =  40           ! sz of pending acts lst
parameter lpae  =   4           ! lngth of pal entry
parameter nlines = 14           ! nr of lines in network
parameter nimps =   6           ! nr of IMPS  in network
parameter nhosts = 20           ! nr of network hosts
parameter crdtty =  5           ! unit for opr crd input
parameter sumtty =  6           ! unit for Summary TTY
parameter disk   =  1           ! disk i/o unit number
parameter logfil =  2           ! unit # for log file
parameter mail   = 14           ! ipcf interrupt signal
parameter mlen   = 30           ! length of ipcf msg
parameter raw    = "40          ! mask for raw tty i/o
parameter cock   = "177737      ! mask for cocked tty
parameter nvmsg = 12            ! nr of voice output msgs
```

2. GLOBAL DATA STRUCTURES (COMMON BLOCKS)

```
COMMON BLOCK EVINF contains info on event list
common/evinf/ttne,tfne,rtmul,rtinv,events,evlp,nev,
+           stime,ctime,evcode,evprms,evsupp

integer ttne               ! time TILL next event
integer tfne               ! time FOR next event
real rtmul                 ! real time multiplier
real rtinv                 ! inverse of rtmul (1/rtmul)
byte events(evlsz)         ! ordered list of events
integer evlp               ! pointer to next event to add
integer nev                ! nr of events on list
integer stime              ! pgm starttime
integer ctime              ! current time (rel to stime)
byte evcode                ! code for current event
byte evprms(4)             ! subcode or params for event
byte subcod
equivalence (subcod,evprms(1))
byte evsupp(26)            ! supplementary event information
```

COMMON BLOCK NETINF contains tables and variables that represent the current state of the network.

```
common/netinf/ istat,lstat,modlns,vhimp,vhnam,
+              imtbf,lmtbf
byte istat(nimps)         ! imp status tbl
byte lstat(nlines)        ! line status table
byte modlns(4,nlines)     ! network topology tbl
byte vhimp(nhosts)        ! host-imp mapping tbl
byte vhnam(10,nhosts)     ! virtual host names
real imtbf                ! MTBF for an IMP
real lmtbf                ! MTBF for a line
```

COMMON BLOCK PALINF contains info on pending action list

```
common/palinf/pal,pap,repeat
byte pal(palsz)           ! pending actions list
integer pap               ! pointer for pal
logical repeat            ! true when we are repeating
                          ! previous events from pal
```

COMMON BLOCK VOCOUT contains info on voice output msgs

```
common/vocout/voflg,logtty,vomnr,vomid,vomsg
logical voflg             ! true if voice output turned on
integer logtty            ! unit for logger tty
integer vomnr             ! message nr within event
byte vomid(2,nvmsg)       ! code and message number
byte vomsg(8,nvmsg)       ! file names where phoneme
                          ! strings are stored
```

COMMON BLOCK VDATA contains phoneme info for current voice output message

```
common/vdata/vphon,infl,rate,nv
integer*4 vphon(maxv)           ! phonemes to be output.
byte infl(maxv), rate(maxv)     ! inflection and rate
byte nv                         ! nr of phonemes
```

COMMON BLOCK MISC contains general purpose stuff

```
common/misc/adate,atime,lbuf,sttyfd,lttyfd,ttmode
byte adate(12)            ! ascii date
byte atime( 8)            ! ascii time
byte lbuf(80)             ! scratch buffer. only use for
                          ! temporary storage of data.
integer sttyfd            ! file descr for summ tty
integer lttyfd            ! file descr for logger tty
integer ttmode(3)         ! tty modes
```

COMMON BLOCK TTYINF contains global data structure for command handling process

```
common/ttyinf/ chan,ccode,delchr,dellin,corplt
integer chan              ! unix chan for Summ TTY
integer ccode             ! code for current opr cmd
```

```
logical delchr          ! true if char was 'delete chr'
logical dellin          ! true if char was 'delete line'
logical complt          ! true if command line completed

COMMON BLOCK IPCF contains necessary variables and
buffers needed to implement primitive IPCF
common/ipcf/ msg,rcvmsg,spid,dpid,newmsg,nomail
byte msg(mlen)          ! send message buffer
byte rcvmsg(mlen)       ! receive message buffer
integer spid            ! sending PID
integer dpid            ! destination PID
logical newmsg          ! true if new msg has arrived.
logical nomail          ! true when mail ints disabled

COMMON BLOCK ENVIR contains info used by both parent
and child processes in CNCC model. Saved in disk file
envir.dat.
common/envir/parent,ttyin
integer parent          ! pid for CNCC process
integer ttyin           ! pid for TTYIN process
```

3. TABLE OF VALID OPERATOR COMMANDS

```
parameter ncmds = 10             ! nr of opr commands
byte cmds(4,ncmds)            ! table of legal commands

cmds(1,j) - nr of chars in jth command
cmds(2,j) - 1st char of jth command
cmds(3,j) - 2nd char of jth command
cmds(4,j) - 3rd char of jth command


data (cmds(i,1),  i = 1,4)/1,'B',0,0/
data (cmds(i,2),  i = 1,4)/1,'P',0,0/
data (cmds(i,3),  i = 1,4)/1,'S',0,0/
data (cmds(i,4),  i = 1,4)/2,'D','U',0/
data (cmds(i,5),  i =  1,4)/3,'D','E','I'/
data (cmds(i,6),  i = 1,4)/3,'R','E','N'/
data (cmds(i,7),  i = 1,4)/3,'R','E','L'/
data (cmds(i,8),  i = 1,4)/3,'R','E','P'/
data (cmds(i,9),  i = 1,4)/1,'Q',0,0/
data (cmds(i,10), i = 1,4)/1,'L',0,0/
```

APPENDIX C

VOICE INPUT VOCABULARY

NR	UTTERANCE	OUTPUT STRING
Ø	ZERO	Ø
1	ONE	1
2	TWO	2
3	THREE	3
4	FOUR	4
5	FIVE	5
6	SIX	6
7	SEVEN	7
8	EIGHT	8
9	NINE	9
1Ø	STOP THE MODEL	?QY
11	QUICKPRINT REPORT	?REPQ:
12	STATUSREP	?REPS:
13	THRUPUT SUMMARY	?REPT:
14	DAYSUMM	?REPD:
15	DUMP IMP1	?DUIMPSYS88$Ø1$
16	DUMP IMP2	?DUIMPSYS88$Ø2$
17	DUMP IMP3	?DUIMPSYS88$Ø3$
18	DUMP IMP4	?DUIMPSYS88$Ø4$
19	DUMP IMP5	?DUIMPSYS88$Ø5$
2Ø	DUMP IMP6	?DUIMPSYS88$Ø6$
21	RELOAD IMP1	?RELIMPSYS99$Ø1$
22	RELOAD IMP2	?RELIMPSYS99$Ø2$
23	RELOAD IMP3	?RELIMPSYS99$Ø3$
24	RELOAD IMP4	?RELIMPSYS99$Ø4$
25	RELOAD IMP5	?RELIMPSYS99$Ø5$
26	RELOAD IMP6	?RELIMPSYS99$Ø6$
27	CLEAR CRASH IMP1	?BCLEARCRASH$1$
28	CLEAR CRASH IMP2	?BCLEARCRASH$2$
29	CLEAR CRASH IMP3	?BCLEARCRASH3
3Ø	CLEAR CRASH IMP4	?BCLEARCRASH$4$
31	CLEAR CRASH IMP5	?BCLEARCRASH$5$
32	CLEAR CRASH IMP6	?BCLEARCRASH$6$
33	MARK UP LINE1	?LU1$
34	MARK UP LINE2	?LU2$
35	MARK UP LINE3	?LU3$
36	MARK UP LINE6	?LU6$
37	MARK UP LINE7	?LU7$
38	MARK UP LINE8	?LU8$
39	MARK UP LINE14	?LU14$
4Ø	CORRECT CKSUM IMP1	?BHACMEM IMP1$1$

41	CORRECT CKSUM IMP2	?BHACMEM IMP2$2$
42	CORRECT CKSUM IMP3	?BHACMEM IMP3$3$
43	CORRECT CKSUM IMP4	?BHACMEM IMP4$4$
44	CORRECT CKSUM IMP5	?BHACMEM IMP5$5$
45	CORRECT CKSUM IMP6	?BHACMEM IMP6$6$
46	SEND LOCAL FILE	?S
47	DELETE FILE	?DEL
48	RENAME	?REN
49	PRINT DIRECTORY	?PD
5Ø	HOW MUCH DISK SPACE	?PT
51	LOGFILE	CNCC.LOG$
52	TOPOLOGY	CNCC.NET$
53	NCC	1$
54	SOLIS	2$
55	DIA	4$
56	NSH	5$
57	COINS TAS	7$
58	ARPANET GATEWAY	8$
59	TILE	1Ø
6Ø	NDS	13
61	BLACKER FRONT END	14
62	BLACKER TAS	19
63	TTRF	2Ø
64	TESTFILE	TESTFILE$
65	SCRATCHFILE	SCRATCHFILE$
66	IMP 1	1$
67	IMP 2	2$
68	IMP 3	3$
69	IMP 4	4$
7Ø	IMP 5	5$
71	IMP 6	6$
72	LOGIN TO UNIX	MALARKEY<cr>
73	PASSWORD	<UNIX PASSWD>
74	CHANGE DIR COINS	CD COINS<cr>
75	STARTUP THE NCC	CNCC|TEE CHAT<cr>
76	STOP THE SYSTEM	?CY
77	HOST SUMMARY	?REPH:
78	BROADCAST FILE	?B
79	RELOAD	?REL
8Ø	MODEM ZERO	Ø
81	MODEM ONE	1
82	MODEM TWO	2
83	NET STATISTICS	NETSTAT$
84	ESCAPE	<esc>
85	DELETE CHARACTER	<ctrl-h>
86	CANCEL	<ctrl-u>

APPENDIX D

VOICE OUTPUT MESSAGES

INTERNAL NUMBER	VOICE MESSAGE
Ø1Ø1	IMP ## errors corrected.
Ø3Ø1	IMP ## has been reloaded.
1ØØ1	Line ## is up.
1ØØ2	Line ## is looped.
1ØØ3	Line ## is down.
11Ø1	IMP ## is down.
12Ø1	Line ## is down.
14Ø1	IMP ## bad host data checksum.
15Ø1	IMP ## trap condition.
16Ø1	IMP ## crash report.
18Ø1	Line ## is down plus side.
18Ø2	Line ## is down minus side.

LIST OF REFERENCES

1. Elovitz, H. S. and Heitmeyer, C. L., "What is a Computer Network?", National Telecommunications Conference 1974 Record, pp. 1007-1014.

2. COINS II Long Range Plan: FY-80 through FY-85, ANNEX B: COINS Network Control Center (CNCC), COINS Project Management Office, November 1979.

3. Bolt Beranek and Newman Report number 3223, Operating Procedures for the COINS Network Control Center (CNCC), by O. Robert Hess, January 1981.

4. COINS II Long Range Plan: FY-81 through FY-85, Annex D1: Network Monitoring Subsystem, COINS Project Management Office, February 1980.

5. Informal correspondence from Mr. George Hicken, COINS Project Manager, August 1981.

6. COINS Project Management Office, "NCC Monitor Report", unpublished report prepared by Mark Herner, CNCC, August 1981.

7. Barr, D. R., Poock, G. K., and Richards, F. R., Experimental Manual Part I: Experimental Methodology, Department of Operations Research Naval Postgraduate School, January 1980.

8. Naval Postgraduate School Report NPS-55-80-016, Experiments with Voice Input for Command and Control, by G.K. Poock, April 1980.

9. Jay G. T., An Experiment in Voice Data Entry for Imagery Interpretation Reporting, Masters Thesis, Naval Postgraduate School, Monterey, California, March, 1981.

10. Martin, T. B., "Practical Applications of Voice Input to Machines", in Dixon, N. R. and Martin, T. B., Automatic Speech and Speaker Recognition, IEEE Press, 1978.

11. Reddy, D. R., "Speech Recognition by Machine: A Review", in Dixon, N. R. and Martin, T. B., Automatic Speech and Speaker Recognition, IEEE Press, 1978.

12. Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R., "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", Computing Surveys, Vol. 12 No. 2, June 1980.

13. Doddington, G. R. and Schalk, T. B., "Speech Recognition: Turning Theory to Practice", IEEE Spectrum, September 1981, pp. 26-32.

14. Cognitronics Corporation, "Voice Response Application Brief: Computerized Telecommunications Switching Systems", 1980.

15. Cognitronics Corporation, "Voice Response Application Brief: Ford Direct Order Entry System", 1980.

16. Simpson, C. A., "Synthesized Voice Approach Callouts for Air Transport Operations", October 1980.

17. Flanagan, J. L., Speech Analysis, Synthesis, and Perception, Springer-Verlag, 1972.

18. Flanagan, J. L., "Computers that Talk and Listen: Man-Machine Communication by Voice", in Dixon, N. R. and Martin, T. B., Automatic Speech and Speaker Recognition, IEEE Press, 1978.

19. Masters, P., "Talk Is Getting Cheaper", Datamation, August 1981, pp. 71-74.

20. Thompson, K. and Ritchie, D. M., Programmers Manual for the UNIX Operating System: Sixth Edition, Revised by Bolt Beranek and Newman Inc., December 1980.

21. Gordon, G., System Simulation, Prentice Hall, 1978.

22. Teja, E. R., "Special Report on Voice Input Technology", EDN, May 27, 1981, pp. 101-113.

23. Threshold 600 User's Manual, Threshold Technology Inc., 1978.

24. SLC-II User's Manual, Digital Pathways, Inc., 1981.

25. Simpson, C. A., SPEECH WIZARD Speech Editor Software, Psycho-Linguistic Research Associates, 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Superintendent ATTN: Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Superintendent ATTN: Chairman, Code 39 C3 Academic Group Naval Postgraduate School Monterey, California 93940	1
4.	Superintendent ATTN: C3 Curriculum Office, Code 39 Naval Postgraduate School Monterey, California 93940	1
5.	Superintendent ATTN: Professor G.K. Poock, Code 55Pk Naval Postgraduate School Monterey, California 93940	10
6.	Superintendent ATTN: Professor A. Andrus, Code 55As Naval Postgraduate School Monterey, California 93940	1
7.	Director National Security Agency ATTN: Mr. George Hicken COINS Project Management Office Fort George G. Meade, Md. 20755	1
8.	Office of the Undersecretary of Defense (R&E) ATTN: Cdr Paul Chatelier Room 3D129, Pentagon Washington, D.C. 20301	1
9.	Naval Electronics Systems Command ATTN: Mr. Frank Deckelman, Code 613 2511 Jefferson Davis Highway Arlington, Virginia 20360	1

10. Wayne Lea 1
889 Sanford Court
Santa Barbara, California 93111

11. Office of Navy Research 1
ATTN: M. Talcott, Code 455
800 North Quincy Street
Arlington, Virginia 22217

12. Threshold Technology, Inc. 1
ATTN: Thomas B. Martin
1829 Underwood Blvd.
Delran, New Jersey 08075

13. Naval Electronics Systems Command 1
ATTN: R. Fratilla, Code 613
2511 Jefferson Davis Highway
Arlington, Virginia 20360

14. National Bureau of Standards 1
Information-Communications Systems Technology
ATTN: Dave Pallet
A219 Technology Building
Washington, D.C. 20234

15. National Photographic Interpretation Center 1
ATTN: Al Hisor
118316 Washington Navy Yard, BLDG 213
1st and M Streets S.E.
Washington, D.C. 20390

16. Bureau of Radiological Health 1
ATTN: Joseph Gitlin
1901 Chapman Avenue
Rockville, Md. 20857

17. Commanding Officer Naval Intelligence Center 1
ATTN: Mr. Albert Weinrauch/IAIPS
4301 Suitland Road
Washington, D.C. 20390

18. Defense Intelligence Agency 1
ATTN: RSE-2 Col. Schwartz
Arlington Hall Station
Arlington, Virginia 22217

19. Director Defense Communications Agency
ATTN: Ed Sumpter, Code 332
Washington, D.C. 20305

20. Director National Security Agency 1
ATTN: Mr. John F. Boehm, R542
Fort George G. Meade, Md. 20755

21. Director National Security Agency 1
ATTN: Mr. Eliot Schmer, S35
Fort George G. Meade, Md. 20755

22. Director National Security Agency 1
ATTN: Mr. Lyle Verbeeck, T333
Fort George G. Meade, Md. 20755

23. Director National Security Agency 1
ATTN: Mr. Kermith Speirman, DDT
Fort George G. Meade, Md. 20755

24. Director National Security Agency 1
ATTN: Mr. Thomas Hassing, T44
Fort George G. Meade, Md. 20755

25. Director National Security Agency 1
ATTN: Mr. Jim Hoffman, T44
Fort George G. Meade, Md. 20755

26. Director National Security Agency 1
ATTN: Mr. James Ritter, T34
Fort George G. Meade, Md. 20755

27. Joanne B. Kim 1
SMC 2812
Naval Postgraduate School
Monterey, California 93940

28. Commander U.S. Army Signal Center and Fort Gordon 1
ATTN: DCD (Major Eisentrout)
Fort Gordon, Georgia 30905

29. Naval Undersea Systems Center 1
ATTN: Dianne Davis, Code 3522
Newport, Rhode Island 02840

30. Naval Ocean Systems Center 1
ATTN: John Schill, Code 824
San Diego, California 92152

31. National Aeronautics and Space Administration 1
ATTN: Dr. James Vorhees
MS 23902
Moffett Field, California 94035

32.	Psycho-Linguistic Research Associates ATTN: Dr. Carol A. Simpson 2055 Sterling Avenue Menlo Park, California 94025	1
33.	Director National Security Agency ATTN: Thomas R. Malarkey COINS Project Management Office Fort George G. Meade, Md. 20755	5